AFRL-IF-RS-TR-2005-253
Final Technical Report
June 2005

ANALYTICAL TOOLS FOR INVESTIGATING AND MODELING AGENT-BASED SYSTEMS

University of Massachusetts

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. K546

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

STINFO FINAL REPORT

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 074-0188*

1. AGENCY USE ONLY (Leave blank)	**2. REPORT DATE** July 2005	**3. REPORT TYPE AND DATES COVERED** **Final Jul 00 – Dec 04**

4. TITLE AND SUBTITLE ANALYTICAL TOOLS FOR INVESTIGATING AND MODELING AGENT-BASED SYSTEMS	**5. FUNDING NUMBERS** C - F30602-00-2-0597 PE - 62301E PR - TASK TA - 00 WU - 09
6. AUTHOR(S) David Jensen	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) University of Massachusetts Knowledge Discovery Laboratory, Department of Computer Science Computer Science Building 140 Governors Drive Amherst MA 01003-9264	**8. PERFORMING ORGANIZATION REPORT NUMBER** N/A
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Defense Advanced Research Projects Agency 3701 North Fairfax Drive Arlington VA 22203-1714 — AFRL/IFED 525 Brooks Road Rome NY 13441-4505	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER** AFRL-IF-RS-TR-2005-253

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: William E. Rzepka/IFED/(315) 330-2762 William.Rzepka@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT *APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.*	**12b. DISTRIBUTION CODE**

13. ABSTRACT *(Maximum 200 Words)*

This research developed techniques for data analysis of multi-agent systems. The research focused on how to analyze relational data that represent sets of interconnected agents, resources, and locations, as well as the attributes of these objects. Results of the research include new algorithms for knowledge discovery, fundamental discoveries about the challenges and opportunities of constructing statistical models in relational data, software prototypes of the algorithms developed, and a simulation environment for evaluating the techniques.

14. SUBJECT TERMS Artificial intelligence, multi-agent systems, machine learning, knowledge discovery and data mining, relational learning, statistical tools, data analysis	**15. NUMBER OF PAGES** 147
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-280-5500 **Standard Form 298 (Rev. 2-89)**
Prescribed by ANSI Std. Z39-18
298-102

Table of Contents

1. Introduction

1.1 Objective of research

The objective of this project was to develop a radically new class of tools to analyze and model systems for agent-based computing. We built on a foundation of analytic approaches developed within intelligence analysis (Sparrow 1991) and social network analysis (Scott 1991; Wasserman and Faust 1994). At the time we proposed this work, these approaches were almost unknown in computer science, but we conjectured that these approaches would be uniquely suited to help understand the behavior of agent-based computing systems. With these tools, we aimed to assist contractors in the TASK program to inquire into the busy, and sometimes covert, behaviors of agents.

Two key technologies lay at the core of this project: 1) a new representation for experimental data; and 2) new techniques for analyzing those data. Specifically, we used a relational data representation that is common in intelligence analysis and quantitative sociology, but rarely used in statistics and data mining. We conjectured that this representation could capture many of the important aspects of the behavior of agent-based systems — far more of those behaviors than traditional data representations. In addition, we developed new tools to facilitate analysis of data recorded in this representation. Based on our work with a prototype system and our work in analyzing the performance of agent-based systems, this appears to be a particularly promising way to accelerate discoveries in agent-based computing and improve our understanding of current systems.

Our project addressed the TASK research goals of modeling agent behavior and modeling the behavior of agent systems. Our tools produce statistical models that predict behavior and more qualitative characterizations of an agent or agent-based system (e.g., clusters of agents and agent groups that exhibit similar behavior or organization). In addition, we found that this modeling and characterization identified key features of agents that should be represented by any agent creation tool and by agents themselves. A common theme in the history of science is that new tools for observation and analysis often spur fundamental advances. We believe that research in agent-based computing will accelerate if investigators are given better methods to record and analyze the behavior of their agents. Investigators will gain greater insight into intentional behaviors, clearer descriptions of emergent behaviors, and faster identification of pathological behaviors. Perhaps most importantly, they will be better equipped to characterize the fundamental principles by which their systems operate, laying the foundation for a science of agent-based computing.

1.2 Approach

Several small communities of academic researchers have pursued work in analyzing relational data, including inductive logic programming (ILP) (Muggleton 1992), relational learning (Quinlan 1990; Friedman, Getoor, Koller, and Pfeffer 1999), and social network analysis (SNA) (Scott 1991; Wasserman and Faust 1994). In addition, software developers in intelligence analysis and law enforcement have devised several practical systems for visualizing relational data.

The focus of our work differed from, and built upon, this prior work in fundamental ways. First, we built tools to discover statistical patterns in relational data. ILP techniques typically require that patterns be deterministic. Second, our tools assist analysts in exploration of the data with a broad set of tools for querying, statistical modeling, and visualization. Techniques in ILP and relational learning are often completely automated, and techniques from intelligence analysis and law enforcement merely assist with visualization, rather than more quantitative analysis. Finally, our techniques analyze large databases, potentially containing tens of thousands of objects and millions of individual pieces of data. In contrast, techniques from ILP, SNA, and applied software products are only applicable to data that are orders of magnitude smaller.

1.3 Outcomes

This research produced key outcomes in four categories:

1) *New knowledge discovery algorithms* — These techniques include three different statistical models (relational probability trees, relational Bayesian classifiers, and relational dependency networks), as well as a graphical query language (QGraph).

2) *Fundamental discoveries about the challenges and opportunities of constructing statistical models in relational data* — We discovered several ways in which relational data invalidate the assumptions of existing methods for statistical modeling, and we discovered new opportunities that could be exploited by techniques specifically designed for relational data. Specifically, we identified how three common features of relational data (concentrated linkage, autocorrelation, and degree disparity) can bias algorithms that construct statistical models and we identified how a new framework for learning and using statistical models (called collective inference) can significantly improve the accuracy of statistical models of relational data.

3) *Software prototypes of the algorithms developed* — We released multiple versions of Proximity, our software environment for relational knowledge discovery. Each of these versions was made available to DARPA contractors. The final versions (3.0 and 3.1) were made available open-source.

4) *Simulation environment for multi-agent systems that helped evaluate our techniques* — We developed the UMass UAV Simulator using the Breve Simulation Environment developed at Hampshire College. We worked jointly with Lee Spector at Hampshire to evaluate our research using the simulator. Hampshire evolved new UAV controllers within the simulator, and we evaluated the behavior of those simulated UAVs.

Support from this contract was instrumental in achieving each of these outcomes, but this contract was not the sole source of support. The research and development outlined here was also supported by another AFRL contract (F30602-01-2-0566). Support was split approximately equally between the two contracts during the period over which both contracts were active.

Finally, students and staff of the Knowledge Discovery Laboratory were awarded first place in the 2003 KDD Cup competition, the most widely recognized competitive evaluation of technologies for knowledge discovery and data mining. KDL's winning team consisted of Amy McGovern, Lisa Friedland, Michael Hay, Brian Gallagher, Andrew Fast, Jennifer Neville and David Jensen. The UMass team competed in the "open task" (one of four tracks in the 2003 KDD Cup competition) that allowed contestants to define their own analysis tasks. Winners were selected by a panel of judges. The title of the UMass entry was "Exploiting Relational Structure to Understand Publication Patterns in High Energy Physics" (McGovern et al. 2003).

2. Knowledge discovery algorithms

We developed three different statistical models, a graphical query language, and pioneered the application of vertical databases to knowledge discovery.

The statistical models represent the conditional and joint probability distributions of attributes in relational data. The models estimate the probability distribution of attributes on an entity or relationship (e.g., the type of an agent, or the probability of success of an inter-agent request) based on the structure and attributes of surrounding entities and relationships (e.g., the number and types of other active requests involving the two agents). Specifically:

- *Relational Bayesian classifiers* — A relational Bayesian classifier (RBC) is a relational version of the simple Bayesian classifier (Neville, Jensen, and Gallagher, 2003). This classifier builds a conditional model of an attribute based on the attributes of surrounding objects and links. Although the RBC is a simple model, it performs quite well.

- *Relational Probability Trees* — A relational probability tree (RPT) selectively considers attributes of nearby objects and links as well as complex aggregates of these attributes to build a conditional model of an attribute (Neville, Jensen, Friedland, and Hay, 2003). Advantages of this model include a correction for the autocorrelation and degree disparity properties found in many relational data sets as well as ease of model understanding (Jensen & Neville 2002; Jensen, Neville, & Hay 2003).

- *Relational Dependency Networks* — Relational dependency networks (RDNs) estimate the joint distribution of attribute values in relational data by extending dependency networks to a relational setting (Neville and Jensen, 2004). RDN models are a new form of probabilistic relational models that offer advantages over relational Bayesian networks (RBNs) and relational Markov networks (RMNs). Advantages of RDN models include an interpretable representation that facilitates knowledge discovery in relational data; the ability to represent arbitrary cyclic dependencies, including relational autocorrelation; and simple and efficient methods for learning both model structure and parameters

In addition to these statistical models, we developed QGraph, a graphical query language that computes fast matches to flexible, high-level descriptions of relational data patterns (Blau, Immerman, and Jensen, 2002). We also pioneered the use of vertical database technology to knowledge discovery algorithms. The use of this technology allows our algorithms to be implemented in ways that allow them to be orders of magnitude faster than systems hosted on SQL databases for the types of operations needed for effective relational knowledge discovery.

3. Statistics of relational knowledge discovery

In addition to new models, we discovered new aspects of statistical reasoning that affect algorithms for learning in relational data. Specifically, we identified three basic characteristics of relational data sets that affect statistical inferences:

- *Autocorrelation and autocorrelation* — We showed that concentrated linkage (the tendency of some types of objects to be linked to a large number of other objects) and relational autocorrelation (the tendency for the same attribute on related items to have correlated values) can cause learning algorithms to be strongly biased toward certain features, irrespective of their predictive power. We identified these characteristics, defined quantitative measures of their severity, and explained how they produce this bias (Jensen & Neville 2002).

- *Degree disparity* — We showed that degree disparity can lead relational learning algorithms to discover misleading correlations. Degree disparity occurs when the frequency of a relation is correlated with the values of the target variable. In such cases, aggregation functions used by many relational learning algorithms will result in misleading correlations and added complexity in models. We examined this problem through a combination of simulations and experiments (Jensen, Neville, and Hay 2003). We also showed how two novel procedures for hypothesis testing can adjust for the effects of using aggregation functions in the presence of degree disparity.

4. Proximity software

Proximity is a software environment that incorporates a wide variety of the research findings and technologies developed under the contract, including new types of statistical models, new learning algorithms, advances in automated inference, corrections for statistical errors that bias the models constructed by more conventional tools, and the QGraph query language. Proximity provides a convenient platform for research into relational knowledge discovery and practical applications to real-world data. In the final version of software developed under this contract (3.1), Proximity encompasses over 30,000 lines of Java code, including a graphical user interface, a Python-based scripting interface, and data export and import facilities.

The first version of Proximity was released in August 2001. Subsequent versions provided progressively greater capabilities:

- *Version 1.1* (October 2001) — Provided new interface capabilities.

- *Version 1.2* (July 2002) — Significantly improved configurability and simplicity of running Proximity, expanded scripting capabilities and graphic user interface capabilities, added a modular QGraph query processor,

- *Version 2.0* (December 2002) — Introduced algorithms for learning and making inferences with relational probability trees as well as major expansions to the implementation of QGraph.

- *Version 3.0* (April 2004) — Completely re-written to use MonetDB, a fast, open-source vertical database. MonetDB allowed Proximity to be orders of magnitude faster than systems hosted on SQL databases for the kinds of operations needed by relational knowledge discovery. In addition, this version featured a greatly extended QGraph Implementation, a graphical editor supporting interactive creation of QGraph queries, and a browser-style graphic user interface, and XML-based data import. This version was also released under an open-source software license and included an 80-page tutorial.

- *Version 3.1* (September 2004) — Introduced an RPT viewer, the ability to run Python commands interactively from a command-line window in the GUI, and social network analysis tools, as well as many bug fixes and performance enhancements.

5. Multi-agent simulators

The evaluation of our techniques for data analysis virtually required a complex multi-agent system. We needed to collect data from such a system, analyze those data, and then be able to examine the validity of the resulting conclusions. We needed a moderately complex multi-agent system, which exhibited emergent behavior, and which was able to run locally with little effort.

Unfortunately, many of the systems being developed by other TASK contractors were not well-suited to evaluating our techniques. Some of the contractors were focused on single-agent systems. Other contractors were not constructing entire multi-agent systems. Still others were constructing systems that were far too simple to tax the analysis capabilities of our techniques.

However, ongoing research at another TASK contractor (Hampshire College) was nearly ideal for our purposes, and we collaborated actively with Hampshire for the last two years of the contract. During that time, we employed the Breve Simulation Environment (developed at Hampshire College) to develop the UMass UAV Simulator. We then provided the UAV Simulator to Hampshire, within which they developed agents various evolutionary programming techniques. We then analyzed the behavior of the evolved agents using Proximity. This interleaved collaboration proved highly successful for both UMass and Hampshire.

6. Discussion and future research directions

Our work has developed several key algorithms for learning statistical models of relational data, discovered some key characteristics of relational data that can strongly bias the results of naive algorithms, and produced reliable software prototypes. Our work has pointed to several important future directions for work on analysis tools for multi-agent systems:

- *Temporal analysis* — The tools that we developed are useful, but their utility is limited when the behavior to be analyzed is strongly temporal. Our tools are primarily useful for analyzing agent organizations rather than agent activities. This is a key distinction, and one that was not clear when we began the project. The next major frontier for these tools is to simultaneously handle both organizational relations and temporal relations.

- *Support for massive data sets* — Many of the actual multi-agent systems we examined could generate massive amounts of data in very little time. When agents change state, communicate, or alter their organizational relationships frequently, this can produce large numbers of records. While we found that a relational representation was essential for capturing the organization and activities of a multi-agent system, handling large number of these records remains a challenge.

- *Visual analysis tools* — Algorithms for constructing statistical models are an essential tool for understanding large data sets, but visual analysis is also extremely important. Many analysts simply want to look at their data. Tools for visualization of data from multi-agent systems remains a key area for future work.

7. References

Blau, H., N. Immerman and D. Jensen (2002). A visual language for querying and updating graphs. University of Massachusetts Amherst Computer Science Technical Report 2002-037.

Friedman, N., L. Getoor, D. Koller, and A. Pfeffer (1999). Learning probabilistic relational models. Proceedings of the International Joint Conference on Artificial Intelligence. 1300-1307.

Jensen, D. and J. Neville (2002). Linkage and autocorrelation cause feature selection bias in relational learning. Proceedings of the 19th International Conference on Machine Learning

Jensen, D., J. Neville and M. Hay (2003). Avoiding bias when aggregating relational data with degree disparity. Proceedings of the 20th International Conference on Machine Learning.

McGovern, A., L. Friedland, M. Hay, B. Gallagher, A. Fast, J. Neville and D. Jensen (2003). Exploiting relational structure to understand publication patterns in high-energy physics. Winning entry: KDD Cup 2003 Open Task, 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

Muggleton, S. (1992). Inductive Logic Programming. New York: Academic Press.

Neville, J. and D. Jensen (2004). Dependency Networks for Relational Data. Proceedings of The Fourth IEEE International Conference on Data Mining.

Neville, J., D. Jensen and B. Gallagher (2003). Simple estimators for relational Bayesian classifiers. Proceedings of The Third IEEE International Conference on Data Mining.

Neville, J., D. Jensen, L. Friedland and M. Hay (2003). Learning relational probability trees. Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

Quinlan, J.R. (1990). Learning logical definitions from relations. Machine Learning 5: 239-266.

Scott, J. (1991). Social Network Analysis: A Handbook. Sage.

Sparrow, M. (1991). The application of network analysis to criminal intelligence: An assessment of the prospects. Social Networks 13:251-274.

Wasserman, S. and K. Faust (1994). Social Network Analysis. Cambridge: Cambridge University Press.

8. Publications supported under this contract

Notes: Publications are listed in order of publication date. One more publication is expected from this contract: A journal article based on McGovern & Jensen 2003.

Jensen, D. and J. Neville (2001). Correlation and sampling in relational data mining. Proceedings of the 33rd Symposium on the Interface of Computing Science and Statistics.

Blau, H., N. Immerman and D. Jensen (2002). A visual language for querying and updating graphs. University of Massachusetts Amherst Computer Science Technical Report 2002-037.

Jensen, D. and J. Neville (2002). Data mining in social networks. National Academy of Sciences Symposium on Dynamic Social Network Analysis.

Jensen, D. and J. Neville (2002). Schemas and models. Proceedings of the Multi-Relational Data Mining Workshop, 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining

Jensen, D. and J. Neville (2002). Linkage and autocorrelation cause feature selection bias in relational learning. Proceedings of the 19th International Conference on Machine Learning.

Jensen, D. and J. Neville (2002). Autocorrelation and linkage cause bias in evaluation of relational learners. Proceedings of the 12th International Conference on Inductive Logic Programming.

Neville, J. and D. Jensen (2002). Supporting relational knowledge discovery: lessons in architecture and algorithm design. Proceedings of the Data Mining Lessons Learned Workshop, 19th International Conference on Machine Learning.

McGovern, A., L. Friedland, M. Hay, B. Gallagher, A. Fast, J. Neville and D. Jensen (2003). Exploiting relational structure to understand publication patterns in high-energy physics. Winning entry: KDD Cup 2003 Open Task, 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

McGovern, A. and D. Jensen (2003). Identifying predictive structures in relational data using multiple instance learning. Proceedings of the 20th International Conference on Machine Learning

Neville, J. and D. Jensen (2003). Collective classification with relational dependency networks. Proceedings of the 2nd Multi-Relational Data Mining Workshop, 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

Neville, J., D. Jensen and B. Gallagher (2003). Simple estimators for relational Bayesian classifiers. Proceedings of The Third IEEE International Conference on Data Mining.

Neville, J., M. Rattigan and D. Jensen (2003). Statistical relational learning: Four claims and a survey. Proceedings of the Workshop on Learning Statistical Models from Relational Data, Eighteenth International Joint Conference on Artificial Intelligence. (uncredited)

Neville, J. and D. Jensen (2004). Dependency Networks for Relational Data. Proceedings of The Fourth IEEE International Conference on Data Mining.

Neville, J., Ö. Şimşek and D. Jensen (2004). Autocorrelation and Relational Learning: Challenges and Opportunities. Proceedings of the Workshop on Statistical Relational Learning, 21st International Conference on Machine Learning

Appendix A: Project Publications

Correlation and Sampling in Relational Data Mining

David Jensen and Jennifer Neville
{jensen|jneville}@cs.umass.edu
Knowledge Discovery Laboratory
Computer Science Department
Univ. of Massachusetts
Amherst, MA 01003 USA

Abstract

Data mining in relational data poses unique opportunities and challenges. In particular, *relational autocorrelation* provides an opportunity to increase the predictive power of statistical models, but it can also mislead investigators using traditional sampling approaches to evaluate data mining algorithms. We investigate the problem and provide new sampling approaches that correct the bias associated with traditional sampling.

1. Introduction

We are studying how to learn predictive models from relational data with PROXIMITY, a knowledge discovery system for analyzing very large relational data structures. In this paper, we report on issues of correlation and sampling that arise from the special features of relational data. Specifically, we identify a common type of probabilistic dependence in relational data that we call *relational autocorrelation*. We show how this type of autocorrelation can bias evaluations of data mining algorithms that use traditional approaches to sample relational data. We describe and evaluate new sampling procedures that can prevent this bias.

We discuss these findings in greater detail below. First, we review current uses of relational data and describe the object-and-relational-neighborhood (ORN) approach to relational learning. Next, we demonstrate and describe relational autocorrelation. We then describe a family of sampling procedures that can be used to evaluate the effects of relational autocorrelation and give the results of applying those procedures in a complex relational data set. Finally, we briefly describe planned future work.

2. Objects and Relational Neighborhoods

In this paper, we examine how to construct and evaluate classification models in *relational* data. Relational data consist of objects (representing people, places, and things) connected by links (representing persistent relationships among objects). Often, the objects and links are heterogeneous, representing many different types of real-world entities and relationships.

For example, relational data could be used to represent information about the motion picture industry, where objects represent studios, movies, awards, and persons (e.g., actors, directors, and producers) and links represent relationships (e.g., actor-in and remake-of). A schema for one such movie database from the UCI repository is shown in Figure 1.

This type of relational data can be stored in conventional relational databases, or in less structured formats such as graph databases, web page repositories, or semi-structured databases. In the case of PROXIMITY, we have implemented a specialized graph database on top of a conventional relational database engine.

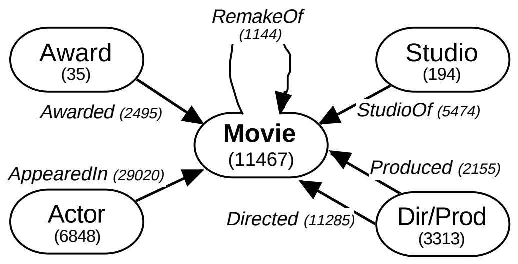


Figure 1*: The movie database schema. Attributes on objects and links are not shown. Numbers indicate the frequency of each element in the UCI movie data.*

Our approach to data mining in relational data focuses on objects and their relational neighborhoods. Our techniques build models that predict the value of an attribute of an object based on *intrinsic attributes* of that object and *relational attributes* that capture characteristics of the "relational neighborhood" of the object. For example, we have constructed statistical models that predict the genre of a movie (e.g., drama, comedy, suspense) based on its intrinsic attributes (e.g., year of production) and its relational attributes (e.g., total number actors). A relational attribute can capture either the characteristics of a single related object (e.g., years of experience of a movie's director) or the aggregate characteristics of multiple objects in the relational neighborhood (e.g., average years of experience of all a movie's actors). We call this approach to relational learning the *object and relational neighborhood* (ORN) approach.

The general idea of ORN is a common feature underlying many systems for relational learning. For example, probabilistic relational models (Friedman et al. 1999) use field values from individual database records and aggregations of values from related records to construct Bayesian networks. WEBKB (Craven et al. 1998) learns naïve Bayes classifiers and rules in first-order logic that employ intrinsic attributes of webpages and their links to other pages to classify the pages into categories (e.g., student homepage, departmental homepage, and course homepage). Several methods from inductive logic programming (Muggleton 1992) use intrinsic and relational clauses to make predictions about individual objects. Finally, our own prior work on PROXIMITY uses an ORN approach to classify objects based on intrinsic and relational attributes (Neville and Jensen 2000). These approaches differ widely in the expressiveness of their model representation and the methods used to construct models. Still, all construct models that predict the attributes of an object based on the attributes of that object and those of closely related objects.

3. An Example: Classifying Movies

We have successfully learned classification models using this basic framework of objects and their local neighborhoods, but relational data can pose unique opportunities and challenges that are not typically encountered in traditional data mining problems. As an example, consider the problem of constructing a classification model to predict the *genre* of a movie (e.g., *drama*, *comedy, suspense*) based on a movie's actors, director, producer, studio, and awards. Specifically, we calculated nine relational attributes of movies (e.g., *AwardCount(x)*, *HasRemake(x)*, *ActorCount(x)*, and *PercentWomenActors(x)*), and used them to construct naïve Bayes classifiers to predict *genre*. We randomly generated five different pairs of training and test sets, where each pair contains disjoint sets of movies.

For each pair, we constructed classifiers on the training set, estimated their accuracy on the test set, and then averaged the five trials.[1]

The classifier is able to achieve an average test-set accuracy of 40%. Accuracy for the best default classifier (*genre = drama*) is 24%, and the standard deviation of the estimate is less than 0.007. While the classifier is far from perfect, it appears to represent a genuine association between the attributes and the class label. If we delve more deeply into the model and analyze the performance of individual attributes, it becomes clear that a model containing only one of the nine attributes can account for nearly all of the elevated accuracy associated with the model. That attribute — *DirectorLastName* — is unusual because it contains such a large number of values. Indeed, individual values sometimes refer to only a single object in the data (e.g., Alfred Hitchcock).

This information about the naïve Bayes model raises an important question — What knowledge does this simple model express?

One way of characterizing the observed behavior is that it represents a relational version of the "small disjuncts" problem (Holte, Acker, & Porter 1989). This problem arises in non-relational learning when overfitted models parse the instance space into sets ("disjuncts") containing only a few data instances. For example, when a decision tree is pathologically large, its leaf nodes can apply to only a single instance in the training set. Such models perform as lookup tables that map single instances to class labels. They achieve very high accuracy on training sets, but they generalize poorly to test sets, when those test sets are statistically independent of the training set.

In the relational version of the small disjuncts problem, models use relational attributes to parse the space of *peripheral* objects into sets so small that they can uniquely identify one peripheral object (e.g., a single Director). If that object is linked to many core objects of a single class (e.g., *genre = comedy*), then a model that uniquely identifies that object can perform well on the training data. If that peripheral object appears in both training and test data, then the model can also perform well on test data.

These conditions certainly hold in the movie data. Single director are often linked to multiple movies, directors often direct movies in a small number of movie genres, and each value of *DirectorLastName* often identifies no more than one director. For example, Table 1 shows the frequency distributions of *genre* for directors linked to more than 25 movies. The distributions clearly differs by director. Ingmar Bergman has directed 28 movies in the data set, and 26 were dramas. Ninety-eight percent of all films directed by Robert Stevens are in the genre of suspense.

We intentionally included *DirectorLastName* in this experiment to evaluate the effect of small disjuncts when classifying movies, although we could have tested the effect by learning classification trees or rules which can use logical combinations of several attributes having fewer values than *DirectorLastName*. In either case, an overfitted model can uniquely identify directors, and then exploit that information when a director appears in both the training and test sets.

However, viewing the observed behavior of the movie classifier as merely the result of the *small disjuncts* problem ignores an important feature of the movie data. The attribute *genre* exhibits *relational autocorrelation.* That is, when objects *x* and *y* are "closely related", then:

[1] We avoided the more traditional approach of 10-fold cross-validation for reasons explained in section 5.

$$p(g(x) \mid g(y)) \neq p(g(x))$$

where g(x) represents the value of some attribute on *x*. In the case of *genre* in the movie data, "closely related" refers to the linkage of two movies through a common director.

Relational autocorrelation is similar to temporal autocorrelation that arises in many econometric problems and other time-series prediction tasks, and it is similar to spatial autocorrelation which arises in statistical models in geography and computer vision.

Relational autocorrelation is easy to demonstrate in the movie data. An expanded version of the contingency table shown in Table 1 can be constructed and analyzed with traditional statistical significance tests. The relationship between genre and director can be tested under the null hypothesis of independence. The relationship is significant at less than the 0.1% level.

Name	ActAdv	Comd	Dram	HistDoc	Other	Rom	ScFiF	Susp	Total
A. Konigsberg	0	15	3	0	0	7	0	0	25
A. Hiller	1	7	4	1	0	1	0	18	32
E. Lubitsch	0	10	9	0	4	9	0	0	32
F. Strayer	0	28	0	0	0	0	1	0	29
G. Thomas	0	30	0	0	0	0	0	0	30
G. Cukor	0	6	12	1	4	4	1	1	29
H. Daugherty	0	0	0	0	0	0	0	28	28
I. Bergman	0	0	26	0	0	1	1	0	28
J. Huston	6	3	8	2	0	2	0	4	25
M. Kertesz	8	1	4	3	5	5	1	3	30
P. von Henreid	0	0	1	0	0	2	0	29	32
R. Altman	1	6	7	3	1	0	2	5	25
R. Stevens	0	0	0	0	0	0	1	49	50
R. Stevenson	2	3	3	1	0	0	9	7	25
S. O'Fearna	12	3	8	3	0	1	0	0	27
A. Hitchcock	2	0	7	0	0	3	0	65	77

Table 1: Frequency distributions of movie genre for all directors linked to 25 or more movies. The distributions differ significantly by director ($p<0.001$).

Autocorrelation is a very common feature of relational data sets. For example, we have observed relational autocorrelation among movie genre with respect to other objects (e.g., studios). We have also observed autocorrelation in a data set on the industry structure of the chemical and banking industries formed from SEC records. Objects in that data set represent companies, officers, directors, owners, as well as ancillary companies such as auditing firms, legal firms, and stock transfer agents. In the SEC data, industry sector exhibits autocorrelation with respect to officers and directors, indicating that persons who sit on multiple boards of directors tend to stay within either banking or chemicals. Autocorrelation was not observed with respect to auditing firms, indicating that the relatively small number of auditing firms tend to target companies in all industries equally.

Relational autocorrelation can be exploited to boost the predictive accuracy of predictive models. For example, our own work has explored *iterative classification*, a technique that builds models based on relational attributes that aggregate predicted class labels of related objects (Neville and Jensen 2000). Iterative classification can be used to identify the predominant genre of a director from information in the test set alone, and then use that information to improve the accuracy of models of a movie's genre. Work by Slattery and Mitchell (2000) explores other methods of explicitly accounting for shared objects in classification of relational data sets. These methods allow classifiers to represent the valid statistical regularities in relational data (e.g., that directors tend to stay within a genre), rather than erroneously concluding that a particular last name (or a unique collection of other attributes) predisposes a director to a genre.

4. Sampling to Detect Autocorrelation

Given that autocorrelation is an important issue in applying data mining algorithms to relational data, our evaluation techniques should be able to test for the existence of relational autocorrelation, and to remove the effects of such autocorrelation when desired. This would allow investigators to clearly understand what effects are responsible for the performance of a given model, and thus understand what features are necessary in new data for models to perform well. For example, if a model to predict movie *genre* relies on *DirectorLastName* and relational autocorrelation with respect to director, then a model learned from movies in the 1950s is unlikely to work well on data from the 1990s, because few directors worked in both decades. In contrast, the model could be expected to perform well on randomly selected data from the early 1960s.

One approach to detecting relational autocorrelation is to test all possible subsets of objects and links that could form connections among the objects being classified. This approach would be a natural one for investigators familiar with data mining, but it has several drawbacks. First, such an approach could be prohibitive when data contain hundreds of possible attributes and many possible subsets of objects through which relational autocorrelation could occur. More importantly, it only allows us to determine that relational autocorrelation exists, but does not allow us to determine the effect on the accuracy of a predictive model nor how the model would perform in the absence of autocorrelation.

The remainder of the paper describes and evaluates an alternative approach — sampling relational data in ways that allow some types of relational autocorrelation and eliminate other types. These techniques allow investigators to train and test of models with the effects of particular types of relational autocorrelation entirely removed, and thus check for specific types of autocorrelation.

Sampling Relational Data

Nearly every data mining algorithm is applied to samples of data drawn from a much larger population of possible instances. Even after one such sample exists, a sampling algorithm is often applied again to further divide the sample into training and test sets for algorithm evaluation, to conduct n-fold cross-validation internal to algorithms, to examine algorithm performance as sample size is varied (Oates and Jensen 1998), or to improve the computational efficiency of a learning algorithm (Srinivasan 1999).

Most sampling in machine learning and data mining assumes that instances are independent and identically distributed (iid). A few areas of data mining explicitly use non-random sampling, but either this work is relatively unusual (e.g., active learning and heterogeneous uncertainty sampling (Lewis and Catlett 1994)) or highly specific (e.g., time series analysis). Most general sampling techniques assume that drawing one instance from a population should not affect the probability of inserting any other instance into that sample.

In contrast, this paper examines situations where instances are not iid, and our samples are created with the ultimate goal of constructing and evaluating predictive models. Methods for sampling relational data are not well understood. In the relatively few cases where researchers have considered special methods for sampling relational data for machine learning and data mining, they have often relied on special characteristics of the data. For example, some researchers have exploited the presence of naturally occurring, disconnected subsets in the population, such as multiple websites without connections among the sites (e.g., Craven et al. 1998). We wish to evaluate classification models that operate over completely connected graphs. There is also a small body of literature on sampling in relational databases (e.g., Lipton et al. 1993), but this work is intended to aid query optimization while our interest is to facilitate evaluation of predictive models.

A sampling algorithm for relational data assigns core and peripheral objects to samples. Recall that core objects are the objects being classified and peripheral objects are all objects needed to calculate relational attributes for a core object. One core object and all its peripheral objects form a *subgraph*. When a sample completely contains a subgraph, the subgraph *j* of sample *k* is denoted S_{jk} (the sample subscript is omitted when not necessary for clarity). An object *i* is denoted O_i. The predicate $in(O_i,S_j)$ indicates that object *i* is contained in subgraph *j*. The predicate $core(O_i,S_j)$ indicates that object *i* is the core object of subgraph *j*, and $periphery(O_i,S_j)$ indicates that object *i* is in the periphery of subgraph *j*. For any given subgraph, the core and periphery are mutually exclusive, and every object in a given subgraph is either in the core or the periphery. Thus, for all *i,j*:

$$core(O_i, S_j) \Leftrightarrow \neg periphery(O_i, S_j)$$
$$in(O_i, S_j) \rightarrow core(O_i, S_j) \vee periphery(O_i, S_j)$$

Note that an individual object can appear as the core of more than one subgraph, as a peripheral object of more than one subgraph, or as the core of some subgraphs and as a peripheral object of other subgraphs.

An algorithm may, or may not, use knowledge about the subgraph membership of objects when constructing the samples. Below, we analyze one technique that ignores subgraph membership and three others that use subgraph membership to decide which objects to add to a given sample. We evaluate each in terms of how the estimate accuracy in the face of relational autocorrelation. We also evaluate their *sampling efficiency* — the fraction of all instances that can be placed into one of several disjoint samples. In random sampling of iid data, sampling efficiency is always 100% (all instances can be placed into one of the samples). In non-random sampling algorithms, sampling efficiency can be lower than 100%. For example, in a population of 50 women and 50 men, a stratified sampling technique that draws a sample containing 66% women (e.g., 50 women and 25 men) would have only 75% sampling efficiency. As we show below, the efficiency of subgraph sampling can also be lower than 100%.

Object Sampling

The simplest technique for sampling relational data is *object sampling* — each object is placed in a sample selected at random, without regard to other objects already in that sample. If two objects are both placed in a given sample, then the links that connect those two objects in the population are also included in the sample. Object sampling is 100% efficient, because all objects are placed in a sample, and it also produces independent samples.

However, object sampling nearly always introduces both bias and variance into the values of relational attributes, and the magnitude of these effects is inversely proportional to the size of samples. These errors arise because object sampling causes subgraphs to be fragmented across multiple samples. Under object sampling, if a given core object is assigned to a particular sample, it is extremely unlikely that all its peripheral objects will be assigned to the same sample. For a given core object, the probability that all peripheral objects will be assigned to the same sample is p^n where *p* is the percentage of all objects assigned to the core's sample and *n* is the total number of peripheral objects for the given core object. Even in extremely favorable conditions (e.g., 50/50 sampling and 5 peripheral objects), the probability that a given core object will have a complete periphery is very small (0.03125). In less favorable conditions (10-fold cross-validation and 25 peripheral objects), the probability of a complete periphery is effectively zero (10^{-25}).

Splitting subgraphs across samples causes *fragmentation effects* in relational data. Some of these effects arise because several techniques for ORN learning rely on relational attributes that apply aggregation functions to attribute values of peripheral objects. For example, relational attributes for movies could include the maximum age of any actor in the movie, the total number of actors, and the mean number of movies made by actors in the movie. Popular aggregation functions include *sum, maximum, minimum, count, mean, median, mode,* and *standard deviation.* These types of aggregation functions are used extensively in probabilistic relational models and in our own learning algorithms for PROXIMITY.

Subgraph fragmentation causes bias and increased variance in aggregate attributes. With some aggregate attributes (those using functions such as *sum, maximum, minimum*, and *count*), fragmented subgraphs leads to statistical bias in attribute estimates. In each case, removing objects from a subgraph can only cause the estimated value of the aggregate attribute to move in a single direction, thus leading to a biased estimate of the true value obtained from the complete subgraph. For example, the aggregate attribute *maximum-actor-age* for a given movie can only remain the same or decrease as peripheral objects (e.g., actors) are removed from the sample containing the core movie. In other aggregate attributes (those using functions such as *mean, median, mode,* and *standard deviation*), a fragmented subgraph leads to increased variance in attribute estimates. In each of these cases, removing objects from a subgraph decreases the number of data points available to estimate these aggregate values, and thus fragmentation increases the variance (conventionally called the *standard error*) of that estimate.

Another fragmentation effect arises because some techniques for ORN learning rely on path attributes that measure the distance from the core object to specific types of peripheral objects. For example, a path attribute might indicate the minimum path length from a core movie to an Oscar-winning movie, traversing the graph through actors and directors. Any movie starring Clark Gable would have a maximum path length of 1, because of his role in the Oscar-winning movie “Gone with the Wind.” An Oscar-winning movie itself would have a path length of zero. Path attributes are not yet extensively used in relational learning, but are common in analyses of social networks (Wasserman and Faust 1994), scientific citations, and the World Wide Web (Watts 1999). Path attributes have also been popularized by concepts such as Erdös number (the number of traversals through coauthored papers necessary to reach from a given person to the mathematician Paul Erdös) and the game “Six Degrees of Kevin Bacon” (where participants attempt to link a given actor to Kevin Bacon by traversals through movies with multiple actors).

Subgraph fragmentation causes positive bias in path attributes. Subgraph fragmentation removes objects (and thus links) from subgraphs, and thus can increase the estimated values path length attributes. Removing a peripheral object either leaves path length unchanged or increases it, and thus subgraph fragmentation can lead to positive bias in estimates of the value of path attributes. In the worst case, subgraph fragmentation can make the estimated path length essentially infinite by making a destination object entirely unreachable.

How do these types of bias and increased variance affect the accuracy of learned models? Bias will cause systematic errors in parameter estimates internal to models (e.g., the split points of decision trees or the probability density estimates in Bayesian nets and naïve Bayesian classifiers). The effect of attribute bias will be particularly pronounced when training and test sets differ in size, because attribute values in each will be biased to differing degrees. For example, object sampling for 10-fold cross-validation would produce training sets with (on average) 90% complete subgraphs, but test sets with subgraphs that are only 10% complete. Increased variance due to subgraph fragmentation will act as a type of attribute noise, which can substantially reduce the accuracy of models (Quinlan 1986). Together, these fragmentation effects indicate that object sampling will produce unrepresentative samples and that alternatives to objects sampling should be strongly considered.

Subgraph Sampling

Subgraph sampling guarantees that entire subgraphs appear within specific samples. Recall that we defined subgraphs as a single core object and all peripheral objects necessary to construct a given set of relational attributes for the core. Sampling entire subgraphs preserves the association between each core object and all the peripheral objects necessary for accurate calculation of attributes.

Table 2 lists a generic algorithm for subgraph sampling. The algorithm first assigns subgraphs to prospective samples, and then incrementally converts prospective assignments to permanent assignments only if the subgraphs are separated from subgraphs already assigned to samples. Several possible definitions of separation are discussed in the next section. This conversion from prospective to permanent samples is done *m* items at a time, where the value of *m* varies by sample to allow creation of samples of unequal sizes.

Table 2: *Algorithm for Subgraph Sampling*

```
Identify all subgraphs $S_i$

Create prospective samples $P_1$, $P_2$,…, $P_n$, and randomly assign each
subgraph to one sample

Create final samples $F_1$, $F_2$,…, $F_n$, and initialize each to the empty
set.

Loop until at least on $P_j$ is empty

    For each prospective sample $P_j$

        Do until $m_j$ subgraphs moved to $F_j$
        or $P_j$ empty

           Select subgraph $S_{ij}$ from $P_j$

           If separate($S_{ij}$,$S_{*k}$) for all $j \neq k$,
           then move $S_{ij}$ from $P_j$ to $F_j$,
           else discard $S_{ij}$ from $P_k$,

Return final samples $F_1$, $F_2$,…, $F_n$
```

Two features of the algorithm are worthy of special note. First, the algorithm enforces separation between subgraphs in different samples to promote statistical independence. As sampling progresses, subgraphs are discarded if they are not separate from subgraphs already assigned to another sample. This feature of the algorithm is intended to prevent specific types of relational autocorrelation among instances in different samples. Three different criteria for subgraph separation are discussed below.

A second feature of the algorithm — the random assignment of subgraphs to "prospective" samples — also promotes statistical independence. The algorithm either makes a prospective assignment permanent, or discards the subgraph. An alternative algorithm would search for an assignment of permanent labels to subgraphs that the maximizes number of subgraphs assigned to each sample. This approach would maximize sampling efficiency (one of the three goals of sampling outlined above).

However, this approach to maximizing sampling efficiency reduces the statistical independence of multiple samples. Consider how such an "optimized" algorithm would behave when confronted with a data set consisting of two disjoint (or nearly disjoint) sets of relational data. Sampling efficiency could be optimized by filling one sample entirely with objects from one disjoint set, and filling another sample with objects from the other set. If the statistical characteristics of one of the disjoint sets did not mirror the characteristics of each other, then accuracy estimates of learned models will be biased downward.

We encountered precisely this problem when sampling relational data about the industrial structure of the banking and chemical industries (Neville and Jensen 2000). Our task was to create classifiers that predicted the industrial sector of a company (*banking* or *chemical*) based on the company's intrinsic and relational attributes. When we created training and test samples for this task, we attempted to optimize sampling efficiency, but we encountered great difficulty obtaining samples with roughly equivalent distributions of the class label. Several of our relational attributes involved companies linked through officers and directors. Officers and directors often link to multiple companies (e.g., an officer of one company and a director of two others), but they almost always stay within a single industry (their area of expertise). As a result, attempting to optimize sampling efficiency harmed statistical independence because it tended to produce training and test samples drawn from different industrial sectors. For this reason, the subgraph sampling algorithm in Table 1 assigns prospective samples randomly and then either confirms that assignment or discards the subgraph. In addition, we stop assigning subgraphs to samples as soon as one of the set of prospective classes has been exhausted.

Subgraph sampling is similar to *snowball sampling*, a sampling technique used in social network analysis (Goodman 1961). In snowball sampling, a single seed object is inserted into a sample, and then all its neighbors are inserted, and then all those objects' neighbors, and so on until a desired sample size (or a specified network boundary) is reached. Snowball sampling was developed for use with *homogeneous* relational data where all objects have the same type (e.g., data where all objects are persons and all links represent acquaintanceship or telecommunications data where all objects are phone numbers and all links are calls). In ORN learning applications, however, objects can have heterogeneous type. Simple snowball sampling could still fragment subgraphs at the boundaries of a sample. Subgraph sampling avoids this fragmentation by accounting for the different roles of core and peripheral objects in attribute calculation.

Subgraph sampling also resembles techniques that construct samples from a small number of completely disconnected graphs. For example, some experiments with WEBKB train classification models on pages completely contained within a single website, and then test those models on pages from another website with no links to the training set. This approach exploits a feature of some websites — heavy internal linkage but few external links. Similarly, some work in ILP constructs samples from sets of completely disconnected graphs (e.g., individual molecules or English sentences). This approach are possible only when the domain provides extremely strong natural divisions in the graphs, and this approach is only advisable where the same underlying process generated each graph. In contrast, subgraph sampling can be applied to data without natural divisions. Where they exist, subgraph sampling will exploit some types of natural divisions. Where they do not exist, logical divisions among subgraphs can be created that preserve the ability to accurately calculate relational attributes and yet preserve the statistical independence among samples.

Determining Subgraph Separation

The algorithm for subgraph sampling (Table 2) depends on the predicate *separate(S_i,S_j)* which indicates whether two subgraphs consist of disjoint sets of objects. We differentiate among three criteria for determining subgraph separation. The criteria are differentiated by the degree to which elements of the two subgraphs are separated. Figure 2 shows the criteria schematically. Informal descriptions and formal definitions are given below.

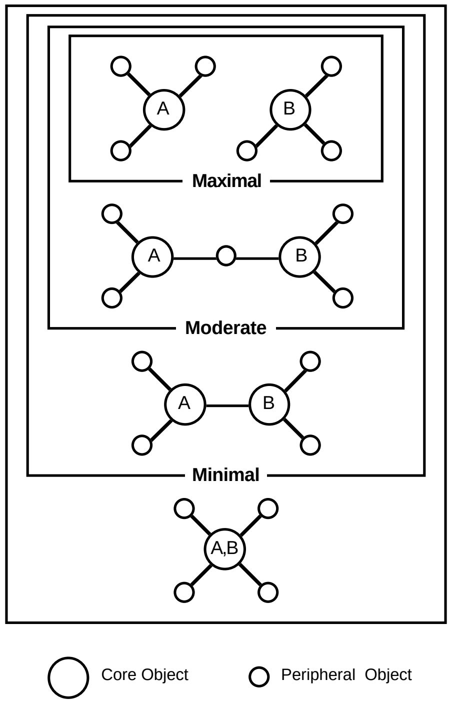


Figure 2: *Three types of subgraph separation. Subgraphs of core objects A and B are in different samples.*

Minimal separation specifies only that subgraphs in different samples have distinct cores, but allows peripheral objects of subgraphs in one sample to appear in the periphery or core of subgraphs in another sample.[1] That is, for all *i*,*m*,*n*,*x*,*y*, where $x \neq y$,

$$core(O_i, S_{mx}) \rightarrow \neg core(O_i, S_{ny})$$

Minimal separation specifies that a single object cannot serve as the core object of a subgraph in more than one sample. This criterion mirrors the constraint on traditional iid sampling, where an instance cannot appear in more than one sample. However, it allows both core and peripheral objects in one sample to appear as peripheral objects in another sample. Minimal separation was used the experiment discussed in Section 2, and thus results obtained using this sampling technique can be affected by relational autocorrelation. That said, minimal separation provides 100% sampling efficiency, because all possible core objects can appear in a sample.

[1] For ease of explanation, we assume that a single object never serves as the core object of two or more subgraphs within a single sample.

Moderate separation adds the condition that the core object of each subgraph is distinct from all peripheral objects in subgraphs of other samples, but still allows peripheral objects of subgraphs in one sample to appear as peripheral objects of subgraphs in another sample. That is, for all *i*,*m*,*n*,*x*,*y*, where $x \neq y$,

$$\begin{gathered} core(O_i, S_{mx}) \rightarrow \neg core(O_i, S_{ny}) \\ core(O_i, S_{mx}) \rightarrow \neg periphery(O_i, S_{ny}) \end{gathered}$$

Moderate separation can provide 100% sampling efficiency if no links directly connect core objects, or if such links exist, but are not used to create relational attributes. For example, the movie data contain almost no direct links between movies (*remake-of* links are the only exception). Thus, judicious creation of relational attributes could allow 100% sampling efficiency even with moderate separation.

Maximal separation adds the condition that the peripheral objects of each subgraph are distinct from the peripheral objects of subgraphs in other samples. Thus, in maximal separation, objects of subgraphs in one sample never appear as objects of subgraphs in another sample. That is, for all *i*,*m*,*n*,*x*,*y*, where $x \neq y$,

$$\begin{gathered} core(O_i, S_{mx}) \rightarrow \neg core(O_i, S_{ny}) \\ core(O_i, S_{mx}) \rightarrow \neg periphery(O_i, S_{ny}) \\ periphery(O_i, S_{mx}) \rightarrow \neg periphery(O_i, S_{ny}) \end{gathered}$$

or, more simply,

$$in(O_i, S_{mx}) \rightarrow \neg in(O_i, S_{ny})$$

In contrast to minimal and moderate separation, there is no sharing of objects among samples and thus there is no opportunity for “information leakage” between the training and test sets. However, maximal separation can exact a heavy toll in terms of sampling efficiency. For populations with even a small number of highly connected objects that are used in relational attributes, this approach can become untenable. For example, the movie data contain a small number of studios that link to hundreds of movies. We explore this problem in more detail in the experiments below.

5. Evaluation

We conducted experiments using the movie data and subgraphs of radius *r=1* (movies and all objects directly linked to them). Using these subgraphs, samples were constructed for each of the three different separation criteria.

Sampling Efficiency

To evaluate the sampling efficiency associated with different separation criteria, we conducted experiments using the movie data. We selected 8,212 movies from the top 20 genres, and created subgraphs of radius *r=1* (movies and all objects directly linked to them) and *r=2* (all *r=1* objects and objects directly linked to them). For each set of subgraphs, we applied subgraph sampling to construct equal-sized training and test sets using each of the three separation criteria.

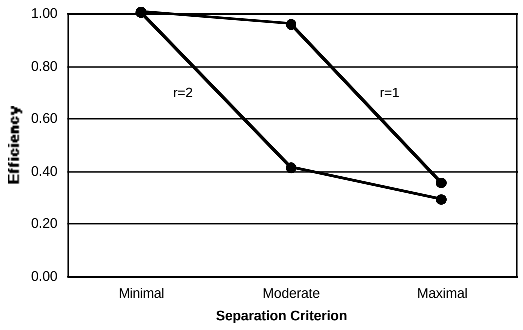


Figure 3: *Sampling efficiency using subgraph sampling with different separation criteria and different-sized subgraphs.*

The sampling efficiency in each of the six cases is shown in Figure 3. Minimal separation is perfectly efficient for the movie data, because no two subgraphs have the same core object. Moderate separation produces only a slight drop in efficiency for *r=1* because two movies that are remakes cannot appear in different samples. As expected, moderate separation for *r=2* and maximal separation for *r=1* and *r=2* produce far larger drops in efficiency.

These results show the best case of sampling efficiency, because subgraphs were placed into only two samples. In cases where a larger number of independent samples are needed (e.g., 10-fold cross-validation), sampling efficiency would be far lower because each subgraph in any one sample would have to be separate from a much larger percentage of all other subgraphs (e.g., 90% in 10-fold cross-validation, rather than 50% in 50/50 sampling).

Relational Autocorrelation

As outlined in the example in Section 2, we constructed classifiers using *DirectorLastName* as the sole attribute. For each pair of training and test sets, we constructed classifiers on the training set and estimated their accuracy on the test set. Figure 4 shows the results for two types of class label. Each point is an average of five trials. Standard deviations are less than 0.01 for minimal and moderate and less than 0.03 for maximal. Accuracy for the best default classifier (*genre = drama*) is 24%. The default accuracy is shown in the figure with a dotted line.

The upper line shows the accuracy of the classifier on the actual class label. Accuracy on test sets generated using *minimal* and *moderate* separation is approximately 40%, but accuracy drops to default when *maximal* separation is used. In samples generated with maximal separation, directors never occur in both training and test sets, and thus the model cannot take advantage of relational autocorrelation.

The lower line in Figure 4 shows the accuracy of the classifier on versions of the data sets with randomized class labels. The class labels for these sets were artificially generated by randomly sampling from the probability distribution of the actual class label. Thus, the randomized class labels are independent of any attribute (including *DirectorLastName)*, and will not exhibit relational autocorrelation. In this case, the classifiers have significantly reduced accuracy when evaluated on samples generated using *minimal* and *moderate* separation.

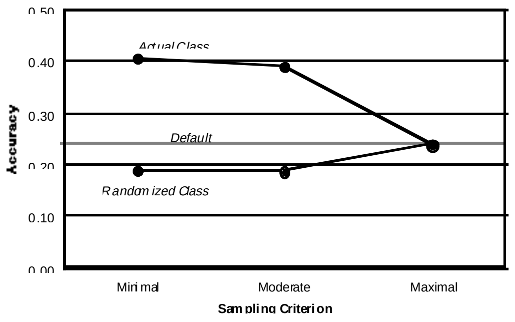


Figure 4: *Estimates of test set accuracy of classifiers using subgraph sampling with different separation criteria. All models use the* DirectorLastName *attribute. Results for predicting the actual class are elevated when samples are generated with* Minimal *and* Moderate *separation. Results for predicting a randomized class label are depressed due to overfitting.*

This effect is due to traditional overfitting. The naïve Bayes classifier estimates *p(class|DirectorLastName)*, and these estimates are often based on very few values. This was not a substantial problem in the non-randomized data, because class distributions were generally highly skewed toward one value. In the randomized case, however, distributions are more uniform. As a result, the classifier sometimes selects the non-default class as the most probable class, thus reducing accuracy. When samples are generated using *maximal* separation, however, the value of *DirectorLastName* in the test set are almost always missing from the model, and the classifier then falls back on its default estimate of the probability distribution over the class labels, and selects the most probable class.

7. Discussion and Future Work

Relational autocorrelation is an intriguing and potentially very useful characteristic of relational data. We are exploring how it can be explicitly incorporated into probabilistic models, learned directly from data, and effectively evaluated. In addition, we are investigating how relational autocorrelation may affect evaluations of existing techniques for learning from relational data. Given that it is such a common feature of relational data sets, relational learning techniques that are prone to overfitting could inadvertently mistake autocorrelation for other types of statistical associations.

Acknowledgments

The authors thank Matt Cornell and Hannah Blau for their work on components of PROXIMITY. This research is supported by DARPA/AFOSR under contract No. F30602-00-2-0597, DARPA/AFRL under contract No. F30602-99-C-0061, and an NSF Graduate Fellowship. The movie data were provided by the UCI KDD Archive. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon. The views and conclusions contained in this paper are solely those of the authors.

References

Craven, M., DiPasquo, D., Freitag, D., McCallum, A., Mitchell, T., Nigam, K., & Slattery, S. (1998). Learning to extract symbolic knowledge from the World Wide Web.

Proceedings of the Fifteenth National Conference on Artificial Intellligence (pp. 509-516). Menlo Park: AAAI Press.

Friedman, N., Getoor, L., Koller, D., & Pfeffer, A. (1999). Learning probabilistic relational models. *Proceedings of the 1999 International Joint Conference on Artificial Intelligence* (pp. 1300-1307). San Francisco: Morgan Kaufmann.

Goodman, L. (1961). Snowball sampling. *Annals of Mathematical Statistics, 32*, 148-170.

Holte, R., Acker, L., & Porter, B. (1989). Concept learning and the accuracy of small disjuncts. Proceedings of the 11th International Joint Conference on Artificial Intelligence (pp. 813-818). Detroit: Morgan Kaufmann.

Lewis, D. & Catlett, J. (1994). Heterogeneous uncertainty sampling for supervised learning. *Proceedings of the 11th International Conference on Machine Learning* (pp. 148-156). San Francisco: Morgan Kaufmann.

Lipton, R., Naughton, J., Schneider, D., & Seshadri, S. (1993). Efficient sampling strategies for relational database operations. *Theoretical Computer Science, 116*, 195-226.

Muggleton, S. (1992). *Inductive Logic Programming*. San Diego: Academic Press.

Neville, J. & Jensen, D. (2000). Iterative classification in relational data. *Papers from the AAAI Workshop on Learning Statistical Models from Relational Data* (pp. 42-49). Menlo Park: AAAI Press.

Oates, T. & Jensen, D. (1997). The effects of training set size on decision tree complexity. Proceedings of The 14th International Conference on Machine Learning (pp. 254-262). San Francisco: Morgan Kaufmann.

Slattery, S. & Mitchell, T. (2000). Discovering test set regularities in relational domains. *Proceedings of the 17th International Conference on Machine Learning*. San Francisco: Morgan Kaufmann.

Srinivasan, A. (1999). A study of two sampling methods for analysing large datasets with ILP. *Data Mining and Knowledge Discovery, 3*, 95-123.

Quinlan, J. R. (1986). The effect of noise on concept learning. In Michalski, R. S., Carbonell, J., and Mitchell, T. M., (eds.), *Machine learning, Vol. II.* Palo Alto: Tioga Press.

Wasserman, S. & Faust, K. (1994). *Social Network Analysis*. Cambridge: Cambridge University Press.

Watts, D. (1999). *Small Worlds*. Princeton, NJ: Princeton University Press.

Data Mining in Social Networks

David Jensen and Jennifer Neville

Knowledge Discovery Laboratory
Computer Science Department, University of Massachusetts, Amherst, MA 01003
{jensen, jneville}@cs.umass.edu

Abstract. Several techniques for learning statistical models have been developed recently by researchers in machine learning and data mining. All of these techniques must address a similar set of representational and algorithmic choices and must face a set of statistical challenges unique to learning from relational data.

Introduction

Recent research projects in two closely related areas of computer science — machine learning and data mining — have developed methods for constructing statistical models of network data. Examples of such data include social networks, networks of web pages, complex relational databases, and data on interrelated people, places, things, and events extracted from text documents. Such data sets are often called "relational" because the relations among entities are central (e.g., acquaintanceship ties between people, links between web pages, or organizational affiliations between people and organizations).[1]

These algorithms differ from a substantially older and more established set of data mining algorithms developed to analyze propositional data. Propositional data are individual records, each of which can be represented as an attribute vector and each of which are assumed to be statistically independent of any other. For example, a propositional data set for learning medical diagnostic rules might represent each patient as a vector of diagnostic test results, and analysis would assume that knowing the disease of one patient tells you nothing about another patient. In contrast, analysis of a relational representation of the same data would retract this latter assumption and add information about familial relationships, workplace contacts, and other relationships among patients that might influence their medical status.

The handful of data mining techniques that have been developed recently for relational data include probabilistic relational models (PRMs) (Friedman, Getoor, Koller, and Pfeffer 1999), Bayesian logic programs (BLPs) (Kersting and de Raedt 2000), first-order Bayesian classifiers (Flach and Lachiche 1999), and relational probability trees (RPTs) (Jensen and Neville 2002). In each of these cases, both the structure and the parameters of a statistical model can be learned directly from data, easing the job of data analysts, and greatly improving the fidelity of the resulting model. Older techniques include inductive logic programming (ILP) (Muggleton 1992; Dzeroski and Lavrac 2001) and social network analysis (Wasserman and Faust 1994).

For example, we have employed relational probability trees (RPTs) to learn models that predict the box office success of a movie based on attributes of the movie and related records,

[1] This meaning of "relational" should be distinguished from the more restrictive meaning of "data stored in relational databases." While relational databases can represent relational data, relational data can also be represented and accessed in other ways.

including the movie's actors, directors, producers, and the studios that made the movie. We have also analyzed relational data in other ways to predict fraudulent cell phone use based on the calling patterns of individual phone numbers. Finally, we have produced models that predict the function and location of proteins in a cell based on network of interactions with other proteins.

Many of these techniques for relational learning share a common set of statistical challenges and design issues. In this paper, we survey these issues, using examples from our work on PROXIMITY, an integrated system for relational learning, and an algorithm for learning RPTs that we have incorporated into PROXIMITY. For each issue, we briefly discuss our design choices in PROXIMITY, and point to alternative approaches used by other systems.

We begin by describing a specific data set and an example analysis task — predicting the box-office receipts of movies — that we use throughout the remainder of the paper. Next, we describe some of the basic features of PROXIMITY and our approach to querying data and learning RPTs. The next two sections discuss a set of representational and algorithmic choices made by the different techniques and a set of statistical issues unique to relational data. We finish with some brief conclusions.

Example Data and Analysis Task

Consider the relational data shown schematically in Figure 1. The data consist of movies and associated objects including people (who act in, produce, or direct the movies), organizations (studios), events (releases of the movie), and other objects (awards). These objects are connected in the ways that you would expect (e.g., actors are linked to movies they act in) and in some occasionally unexpected ways (e.g., movies are linked directly to other movies that are remakes). In addition to the high-level structure of the database shown in Figure 1, the database contains attributes associated with each object, including the titles and genres of movies, the names and ages of persons, and the countries and box-office receipts of movie releases.

The data are drawn primarily from a large online resource, the Internet Movie Database (*www.imdb.com*) that makes its data public for research and other non-commercial purposes. In addition, we have added other data drawn from the Hollywood Stock Exchange (*www.hsx.com*), an artificial market where players trade in stocks that track the relative popularity of movie actors.

The data are voluminous, consisting of over 300,000 movies, 650,000 persons, and 11,000 studios. Those objects are connected by over 2.3 million acted-in links, 300,000 directed links, and 200,000 produced links. The available data on movies vary widely. For example, not all movies have releases, and HSX data are only available for a small percentage of actors in IMDb. Data are more complete for more recent movies and persons.

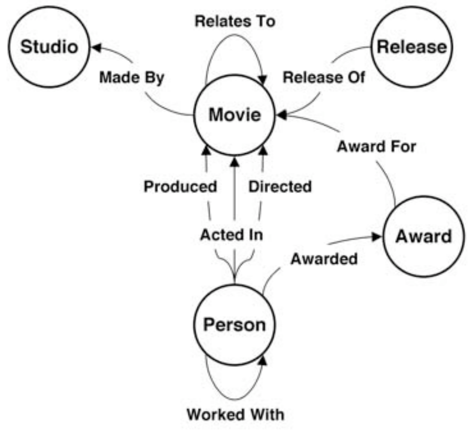


Figure 1: Example schema for data from the Internet Movie Database.

The movie data support a variety of interesting predictive modeling tasks. We have already mentioned one — predicting the opening weekend box office receipts of a movie — and we will use this task as an example throughout the paper. Specifically, we will focus on predicting a probability distribution over a simple binary attribute of movies — Does the movie make more than $2 million in its opening weekend? We will call this attribute *receipts*.

We could attempt to predict other attributes of objects (e.g., a movie's genre or an actor's gender) or attributes of links (e.g, the type of a link between a person and a movie) with PROXIMITY. In addition to these, other types of prediction tasks are certainly possible. One could attempt to learn models that predict missing links between objects. For example, reviewers sometimes call a movie a "crypto-sequel" when it stars the same actors and has a similar plot line as another recent movie, but does not explicitly tie the two storylines. For example, the 1998 movie "You've Got Mail" starring Tom Hanks and Meg Ryan was said to be a crypto-sequel to the 1993 movie "Sleepless in Seattle" (as well as a remake of the 1940 movie "Shop Around The Corner" starring James Stewart and Margaret Sullavan). Given enough examples of crypto-sequels, a data mining algorithm could learn a predictive model from the movie data. Recent work by Getoor, Friedman, Koller, and Taskar (2001) has created models that predict the existence of missing links.

One could also attempt to learn models that predict an attribute of a subgraph, rather than only a single object or link. For example, the emergence of a highly paid Hollywood movie star may consist of a set of successful movies in which the actor had a starring role and one or more awards. Models of this pattern would consist of many objects and links, combined in a particular temporal sequence.

In this paper, we will focus almost exclusively on the task of learning probability distributions over the values of attributes of objects and links. While predicting link existence and classifying subgraphs are extremely interesting problems, the techniques learning probabilistic models for these tasks are much less numerous and much less well-developed than for simple attribute modeling.

One important input to relational learning algorithms is a *schema* or interpretation of the data that specifies a type system over the objects and links in the data. For example, Figure 1 above specifies one schema for the movie data, but others are possible. For example, an alternative schema might specify people as either actors, directors, or producers. Figure 2 provides a hierarchy of possible object types as well as two possible families of schemas constructed from those object types (a full schema would also specify a set of link types). Such a hierarchy is sometimes called an *ontology* (Gruber 1993).

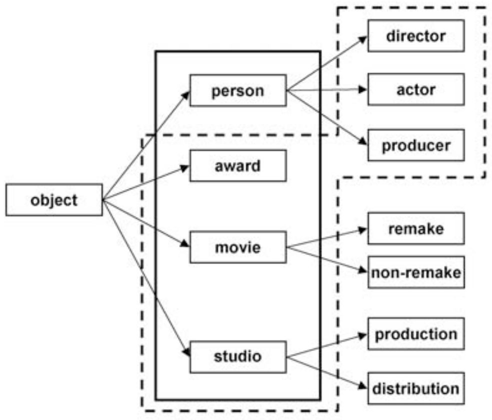


Figure 2: An example ontology of movie objects.

Querying and Learning

To address learning tasks of this kind, our research group is constructing PROXIMITY — a system for machine learning and data mining in relational data. The system is designed as a framework within which a variety of analysis tools can be used in combination. At the foundation of PROXIMITY is a graph database for storing semi-structured data that can be represented as a graph. The database can be accessed by tools for querying data, sampling data, and calculating attributes that depend partially or entirely on network structure (e.g., measures drawn from social network analysis). Sampled data can then be analyzed with tools that construct statistical models. Finally, all these tools can be called from a scripting language interface. In addition to these components, we are developing additional components for clustering, graph partitioning, and additional types of statistical modeling.

In this paper, we will focus on a relatively simple combination of two tools — our query language and one of our learning algorithm. The query language is a visual language for expressing queries to the graph database. The learning algorithm constructs relational probability trees (RPTs), a type of probabilistic classifier for relational data. The two components work in concert. The query language is used to extract subgraphs from a large network of data; the RPT algorithm is used to learn a model that estimates a conditional probability distribution for the value of an attribute of a class of objects or links represented in all those subgraphs. That estimate is conditioned on the attributes of other objects and links in the subgraph.

For example, a query might extract subgraphs consisting of a movie and all directly related actors, producers, directors, studios, and awards. An RPT could then be constructed to estimate the probability that a movie makes more than $2 million in its opening weekend (*receipts* = *True*), given attributes of the actors, producers, directors, studios, and awards. Note that different movies will have different numbers of related objects such as actors and awards. Thus, the subgraphs could not be represented directly as simple attribute vectors.

Our query language, QGraph, represents queries as graphs with associated attribute constraints and annotations on vertices and edges (Blau, Immerman, and Jensen 2002). For example, Figure 3 shows the query described above with a movie and all its related objects. The numeric annotation [1..] on the actor vertex specifies that a match must have one or more actors, and that all associated actors should be returned as part of each matching subgraph. Some object types and link types are left unspecified because of known connectivity constraints in the data. Matches to the query are shown in Figure 4. Actual names of people, studios, and movies are left out for simplicity. The first match has three actors and no award; the second has four actors and no award, and shares an actor and a studio with the first match; the third match has only a single actor, but won an award. The fact that entire subgraphs are returned as part of a match is a subtle, yet vital, feature of the language for our purposes. Other languages such as SQL, for example, can only return a single record as a match, not a record of variable size, such as a subgraph.

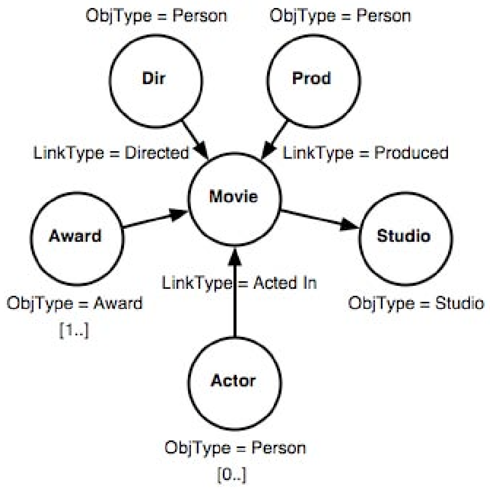


Figure 3: QGraph query for IMDb data.

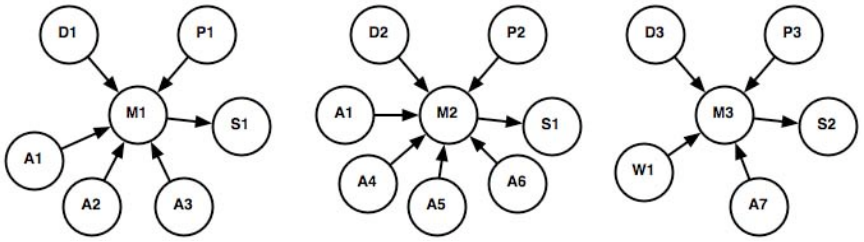


Figure 4: Matches to the query in Figure 3.

Our learning algorithm for relational probability trees constructs trees such as the one shown in Figure 5. The tree represents a series of questions to ask about any subgraph returned by the corresponding query. In this tree, the root node asks whether the movie has more than five actors born after 1943. If so, the subgraph travels down the left-hand branch to a node asking whether the movie at the center of the subgraph is a drama. The subgraph continues moving down appropriate branches of the tree until a leaf node is reached. The leaf nodes contain probability distributions over the values of the *receipts* attribute. Leaf nodes in Figure 5 shows the number of movie subgraphs of each class that reach the leaf, as well as their respective probabilities. The leftmost pair of numbers indicate the number and probability of movies with opening weekend box office receipts exceeding $2 million (*receipts = True*). The second numbers indicate the converse (*receipts = False*).

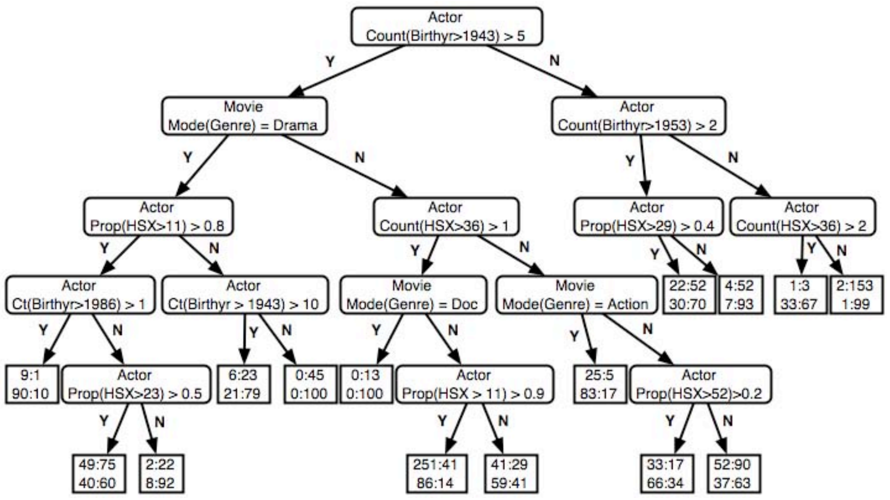


Figure 5: An example relational probability tree (RPT)

Our construction algorithm for RPTs is a recursive partitioning algorithm similar in spirit to CART (Breiman, Friedman, Olshen and Stone 1984), C4.5 (Quinlan 1993), and CHAID (Kass 1980). However, the RPT algorithm searches over the attributes of different object types in the subgraph and multiple methods of aggregating the values of those attributes and creating binary splits on those aggregated values. For example, for a numeric attribute such as birth year, it searches over splits such as MEAN(birthyr) > x, PROPORTION(birthyr > x) > y, MAXIMUM(birthyr) > y, MINIMUM(birthyr) > x, and COUNT(birthyr > x) > y. Our current approach continues partitioning the training data until a stopping criteria is reached. Our current stopping criteria uses a Bonferroni-adjusted chi-square test analogous to that used in CHAID. However, such methods face a variety of problems due to multiple comparison effects (Jensen and Cohen 2000), and we are exploring the use of randomization tests (Jensen 1992) to better adjust for such effects.

This two-step approach of querying and then learning is necessary because of the semi-structured data model that underlies Proximity. In Proximity's graph database, objects and links are not created with strong type information. Rather, data about each object or link is stored in zero or more attributes, name-value pairs such as <age, 54> or <genre, comedy>. Even type information (e.g., person or movie) is stored as an ordinary attribute without privledged status. As a result, attributes are not constrained to occur in particular combinations, in contrast to more conventional relational databases, where a static schema defines both type information and the fields (attributes) corresponding to each entity or relation type. If such structure is needed in Proximity, it can be imposed by a QGraph query. The labels in a query (e.g., the "movie", "actor", and other labels in Figure 3) are assigned to the matching portions of a subquery and remain on those elements for use by other algorithms such as the RPT construction algorithm. Similarly, we often employ particular schemas (such as the one shown in Figure 1) to aid communication, but this is a convenience, not a necessity.

This high degree of flexibility imposes a performance penalty for querying. However, such flexibility is essential for effective machine learning and data mining. First, practical data mining often involves the creation of many new attributes as a human data analyst tries alternative methods for understanding and modeling the data. Adding many attributes to a conventional database would require constant updates to its schema, a costly operation for traditional relational databases. Second, a particular schema is just one way of interpreting a given data set, and it can bias analysis in important ways. To enable truly effective data mining, analysts must be able to change the schema easily, and thus reconceptualize the domain (Jensen & Neville 2002b; Neville & Jensen 2002).

Comparison and Contrast

Techniques for relational learning can be better understood by examining them in the context of a set of design choices and statistical issues. This section describes several decision choices and the next section covers a small set of unique statistical issues facing relational learning algorithms.

Data characteristics

- *Network size* — Raw size is one of the most obvious methods of characterizing a relational data set. PROXIMITY has been constructed and evaluated on relatively large networks. The largest data set we have analyzed (on wireless phone fraud) contains nearly 2 million objects and 7 million links. The complete IMDb data set contains over 1.1 million objects and over 3.1 million links. These fairly large data sets contrast with the relatively small networks typically examined by work in social network analysis and inductive logic programming.

- *Connectivity* — The degree of connectivity among different portions of the data graph is another important characteristic of relational data sets. Our work focuses on networks consisting of a small number of large connected components. In contrast, much of the work in ILP and SNA has focused on many small disconnected components, each of which can be considered a data instance. For example, some work in ILP has analyzed the relational structure of molecules to predict their mutagenicity (Srinivasan, Muggleton, Sternberg, and King 1996). Each molecule is considered a single instance for purposes of learning.
- *Homogeneity* — Many techniques that analyze relational data assume the data consist of homogeneous objects. Such networks include sets of web pages, phone numbers, or persons within an organization. In contrast, several recently developed techniques, including our work on RPTs, can analyze sets of relational data with heterogenous objects, such as movies, people, and studios that make up the IMDb data.

Task

- *Level of relational dependence* — The most commonly used modeling techniques from machine learning, data mining, and statistics analyze independent attribute vectors, thus assuming that relational dependencies are unimportant, or at least beyond the scope of analysis. Specialized techniques for spatial and temporal data have been developed that assume a highly regular type of relational dependence. In contrast, the work discussed here addresses relational data sets with potentially irregular relational structure, with variation in the number and type of links among objects, and these variations are assumed to have significance for modeling.
- *Type of task* — Nearly all the algorithms discussed here focus on *supervised learning*. That is, they attempt to predict the value of some attribute whose true value is known in the data set. In contrast, some approaches focus on *unsupervised learning*, where the task is to discern some unknown structure in the data. Clustering algorithms are a form of unsupervised learning, and similar work has recently been undertaken for relational data (e.g., Taskar, Segal, and Koller 2001).
- *Level of determinism* — RPTs, PRMs, and many of the other approaches discussed here attempt to learn *probabilistic* models of relational data. However, some techniques are specially adapted to learning in deterministic domains. For example, such techniques have been applied to chess, learning grammars for artificial and natural languages, and inducing computer programs from examples. Most work in inductive logic programming is focused on deterministic domains, though some recent work extends this work into probabilistic domains (Dzeroski and Lavrac).

- *Locality of inference* — PROXIMITY's combination of querying for subgraphs and learning based on those subgraphs assumes that all relevant relational information is preserved in the portion of the entire data set represented in the subgraph. If important information resides on elements outside the matched subgraph, then the RPT cannot capture it. The subgraph is assumed to represents the relevant "local neighborhood" of an object (e.g., a movie), and more global features of the graph are assumed to be unimportant. Similar locality constraints apply explicitly or implicitly for most techniques, but the degree of these constraints can vary considerably.

Model Representation and Learning

- *Type of model* — To date, we have incorporated modeling algorithms into PROXIMITY that construct conditional or discriminative models. This contrasts with other work focused on constructing generative models. Generative models define a probability distribution over the entire space of data instances. For example, for the problem of predicting the receipts of movies, a generative model would define the probability of all possible movie subgraphs along with a probability distribution over possible values of the receipts attribute. In contrast, a discriminative model defines a probability distribution over the values of receipts, given a particular subgraph. As with other types of Bayesian network models, PRMs are generative models. As with other types of tree-based models, RPTs are discriminative models. Generative models have a wider range of uses (such as detecting anomalies in a data set), provide a more complete description of the dependencies in a data set, and allow for more robust inference in the presence of missing data. However, their accuracy on purely discriminative tasks is often lower than models explicitly learned for that purpose, and they can be more difficult to learn.
- *Search over model structures* — The RPT learning algorithm searches over a wide range of possible structures for the tree and for the attributes included in the tree. In contrast, some approaches to relational learning, including first-order Bayesian networks, PROXIMITY's own relational Bayesian classifer, and other techniques in social network analysis only learn the parameters for a model with fixed structure and attributes.
- *Attribute construction* — RPT learning involves a limited form of attribute construction. Aggregate attributes (e.g., average actor age) are constructed and evaluated when constructing the tree. Some techniques such as ILP offer far more extensive search of such "constructed" attributes, greatly expanding the set of possible models that can be learned (Silverstein and Pazzani 1991). Other techniques do no search whatsoever, relying on the existing attributes on objects and links.
- *Use of background knowledge* — Data analysts often have substantial background knowledge that can greatly assist model construction. Some techniques can used encoded background knowledge in the learning process. For example, background knowledge in first-order logic can be used by ILP approaches to speed and improve learning. Similarly, prior probability distributions can be used in Bayesian learning techniques. To date, PROXIMITY does not employ any explicit form of background knowledge in its learning algorithms.

Statistical Issues

Our recent work on relational learning has concentrated on the unique challenges of learning probabilistic models in relational data. Specifically, we are examining how particular characteristics of relational data affect the statistical inferences necessary for accurate learning. We have identified three features of relational data — concentrated linkage, degree disparity, and relational autocorrelation — and shown how they lead to two pathological behaviors in learning algorithms.

To explain more fully, the relevant features of relational data are:

- *Concentrated linkage* — Real relational data sets can show striking non-uniformities in the concentration of linkage between different types of objects. For example, in our IMDb data, movies are linked to only a single primary studio, and each such studio is typically linked to many movies. We refer to this as *concentrated linkage* (Jensen and Neville 2002a). It contrasts with other situations where a smaller number of movies link to a single object (e.g., directors) or where many movies link to many objects of the same type simultaneously (e.g., actors). Figure 6 shows a schematic of the two situations. We have found concentrated linkage in many relational data sets. Perhaps the best example is publicly traded companies that each link to a single accounting firm, of which there are only a very small number.

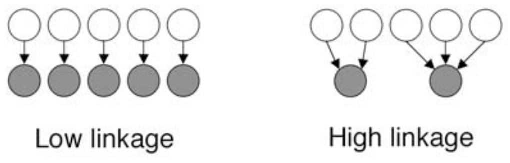


Figure 6: Concentrated linkage

- *Degree disparity* — Another characteristic that occurs in some relational data sets is *degree disparity*. This condition arises when objects of different classes have widely different distributions of degree (the number links to objects of a particular type). For example, in IMDb, we found that US-based studios were systematically linked to a larger number of movies than foreign studios ($p<0.0001$). Figure 7 shows degree disparity schematically. We have found similar degree disparity in other data sets. For example, the number of owners differs systematically among publicly traded companies in different industries and the number of hyperlinks differs systematically among different classes of web pages at university web sites.

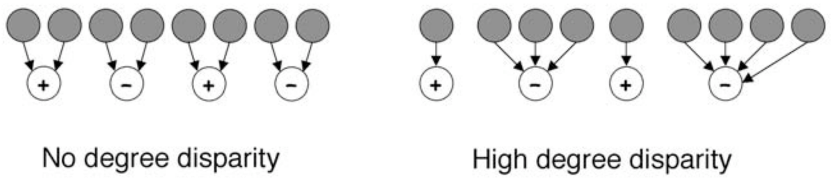


Figure 7: Degree disparity

- *Relational autocorrelation* — Autocorrelation is the correlation among values of the same attribute for related objects. For example, temporal autocorrelation occurs when values of a given attribute (e.g., stock price) at time t tend to correlate highly with the value of the same attribute at time t+1. By analogy, we define *relational autocorrelation* as the correlation among values of given variable on objects that are nearby in graph space (Jensen and Neville 2002a). For example, the box office receipts of a movie tend to be highly correlated with the receipts of other movies made by the same director (correlation coefficient = 0.65) but not for movies starring the same actors (correlation coefficient = 0.17). Figure 8 shows autocorrelation schematically. We have found many other examples of autocorrelation, including correlation of the fraud status of interconnected wireless phone numbers and topics of interconnected web pages.

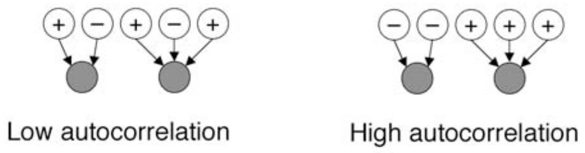


Figure 8: Relational autocorrelation

These three characteristics of relational data can greatly complicate efforts to construct good statistical models. Specifically, they can lead to:

- *Biased feature selection* — Our recent work has shown that high levels of concentrated linkage and relational autocorrelation can cause data mining algorithms to select models that have the weakest, rather than the strongest, statistical support from the data (Jensen and Neville 2002a). This pathology occurs because linkage and autocorrelation combine to reduce the effective sample size of the data, thus increasing the variance of statistics used to assess the relatively utility of different components in learned models. Given that learning algorithms select the best component among many options, they can often select components with high variance, but low true utility, thus reducing the overall accuracy of the resulting model.

- *Spurious correlation* — In other work, we demonstrate a pathology associated with building models that aggregate the values of many objects (e.g., the ages of many actors associated with a movie). This is a common method for simplifying relational data, and it is used in both RPTs and PRMs. When aggregation is used on data with degree disparity and autocorrelation, it can lead data mining algorithms to include completely spurious elements in their models (Type I errors) and to completely miss very useful elements (Type II errors) (Jensen and Neville, in preparation). These errors occur with degree disparity because many aggregation functions (e.g., Max) will produce apparent correlation between the aggregated values (e.g., maximum movie receipts) and a class label (e.g., studio location) whenever degree disparity occurs, regardless of whether movie receipts has any correlation with studio location.

Both of these effects show the problems associated with violating the assumption of independence among data instances that underlies so many of the techniques common to machine learning, data mining, and statistical modeling techniques. These results imply that new approachhes are necessary to extend current techniques for data mining to relational data. We are developing one potentially promising class of techniques, based on randomization tests and resampling-based methods. We expect that these computationally intensive statistical procedures will allow us to adjust for the unique characteristics of a given relational data set, and make accurate parameter estimates and hypothesis tests. We are incorporating these approaches into our algorithm for constructing relational probability trees. We conjecture that similar approaches will need to be incorporated into all accurate techniques for building statistical models from relational data.

Conclusions

Recent work in machine learning and data mining has made impressive strides toward learning highly accurate models of relational data. However, little of this work has made good use of research in other areas, such as social network analysis and statistics. Cross-disciplinary efforts and joint research efforts should be encouraged to promote rapid development and dissemination of useful algorithms and data representations. In particular, this work should focus on the unique statistical challenges raised by relational data.

References

Blau, H., N. Immerman, and D. Jensen (2002). A Visual Language for Querying and Updating Graphs. University of Massachusetts Amherst Computer Science Technical Report 2002-037.

Breiman, L., J. Friedman, R. Olshen, and C. Stone (1984). *Classification and Regression Trees.* Belmont, CA: Wadsworth International Group.

Dzeroski, S. and N. Lavrac, (Eds.) (2001). *Relational Data Mining.* Berlin: Springer.

Flach, P. and N. Lachiche (1999). 1BC: A first-order Bayesian classifier. Proceedings of the Ninth International Workshop on Inductive Logic Programming (ILP'99). S. Dzeroski and Peter Flach (Eds.). Springer. 92-103.

Friedman, N., L. Getoor, D. Koller, and A. Pfeffer (1999). Learning Probabilistic Relational Models. In IJCAI'99. 1300-1309.

Getoor, L., N. Friedman, D. Koller, and B. Taskar (2001). Learning probabilistic models of relational structure. Proceedings of the Eighteenth International Conference on Machine Learning (ICML).

Gruber, T. (1993). Towards principles for the design of ontologies used for knowledge sharing. *Formal Ontology in Conceptual Analysis and Knowledge Representation.* Kluwer Academic Publishers. N. Guarino and R. Poli (Eds.).

Kass, G.V. (1980). An exploratory technique for investigating large quantities of categorical data. *Applied Statistics* 29:119-127.

Kersting, K. and Luc De Raedt (2000). Bayesian logic programs. Proceedings of the Work-in-Progress Track at the 10th International Conference on Inductive Logic Programming. J. Cussens and A. Frisch (Eds.). 138-155.

Jensen, D. (1992). Induction with Randomization Testing. PhD thesis. Washington University. St. Louis, Missouri.

Jensen, D. and P. Cohen (2000). Multiple comparisons in induction algorithms. *Machine Learning* 38: 309-338.

Jensen, D. and J. Neville (2002a). Linkage and autocorrelation cause feature selection bias in relational learning. Proceedings of the 19th International Conference on Machine Learning.

Jensen, D. and J. Neville (2002b). Schemas and models. Proceedings of the Multi-Relational Data Mining Workshop, 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

Jensen, D. and J. Neville (in preparation). The effect of degree disparity on feature selection in relational learning.

Muggleton, S. (Ed.) (1992). *Inductive Logic Programming.* San Diego, CA: Academic Press.

Neville, J. and D. Jensen (2002). Supporting relational knowledge discovery: Lessons in architecture and algorithm design. Proceedings of the Data Mining Lessons Learned Workshop, 19th International Conference on Machine Learning.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning.* San Mateo, CA: Morgan Kaufmann.

Silverstein, G. and Pazzani, M. (1991). Relational cliches: Constraining constructive induction during relational learning. Proceedings of the Eighth International Workshop on Machine Learning.

Srinivasan, A., S. Muggleton, M. Sternberg, R. King (1996). Theories for mutagenicity: A study in first-order and feature-based induction. *Artificial Intelligence* 85: 277-299.

Taskar, B., E. Segal, and Daphne Koller (2001). Probabilistic classification and clustering in relational data. Proceeding of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01). 870-878.

Wassermann, S. and Faust, K (1994). *Social Network Analysis: Methods and Applications*. Cambridge: Cambridge University Press.

Acknowledgments

This research is supported by the Defense Advanced Research Projects Agency under contract numbers F30602-01-2-0566 and F30602-00-2-0597, respectively. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements either expressed or implied, of DARPA or the U.S. Government.

Schemas and Models

David Jensen and Jennifer Neville

Computer Science Department, University of Massachusetts, Amherst, VA 01003

{jensen, jneville}@cs.umass.edu

Abstract. We propose the *Schema-Model Framework*, which characterizes algorithms that learn probabilistic models from relational data as having two parts: a schema that identifies sets of related data items and groups them into relevant categories; and a model that allows probabilistic inference about those data items. The framework highlights how relational learning techniques must structure their own learning tasks in ways that propositional learners do not. The framework also highlights interesting directions for future research in relational learning.

1 Introduction

Several techniques have been developed that learn probabilistic models from data with complex relational structure. These include techniques for learning probabilistic relational models (Friedman, Getoor, Koller, and Pfeffer 1999), first-order Bayesian classifiers (Flach and Lachiche 1999), and relational probability trees (Neville and Jensen 2002). The number of techniques has grown to the point that they provide a sufficient basis for generalizations about some aspects of learning probabilistic models from relational data.

In this paper, we do three things. First, we describe the general characteristics of the learning task addressed by these techniques. We introduce this characterization only to provide a foundation for the second portion of the paper. In this second portion, we propose the *Schema-Model Framework*, which characterizes algorithms that learn probabilistic models from relational data as having two parts— a *schema* that identifies sets of related data items and groups them into relevant categories; and a *model* that allows probabilistic inference about those data items. Third, we discuss several new research problems that arise in the context of schemas and models.

We propose the Schema-Model Framework (SMF) to highlight some of the unique challenges and opportunities of learning from relational data, rather than to provide an overarching characterization of all aspects of these learning algorithms. The SMF embodies a relatively simple point— techniques that learn from relational data must structure their own learning tasks in ways that propositional learners do not. The SMF helps identify: 1) new representational and algorithmic elements that are needed to learn from relational data; 2) alternative methods used by learning techniques to implement these elements; and 3) new directions for research in relational learning.

D. Jensen and J. Neville (2002). Schemas and models. *Papers of the KDD 2002 Workshop on Multi-Relational Data Mining.*

The next two sections lay the groundwork for discussing the SMF. These sections can be skimmed or skipped entirely by readers familiar with techniques for learning from relational data. Section 4 introduces the framework, and section 5 presents some new learning problems that it highlights. The final section discusses some related and future work.

2 Probabilistic relational learning

In this section, we attempt to provide a general characterization of learning probabilistic models from relational data. For the remainder of the paper, we shorten this to "relational learning" where the reference to probabilistic learning is clear from context.

2.1 Data

A data set D consists of a set of *objects* and *links*, $D = \{O,L\}$. Objects o_i { O generally represent people, places, things, and events. For example, in the domain of movies, a data set might contain objects representing movies, actors, directors, producers, studios, and movie releases.[1] Links l_i { L represent relations between two or more objects. For example, in the movie domain, a link might represent the relation *Directed(Director,Movie)*, *Produced (Producer,Movie)*, or *Awarded(Award,Actor,Movie)*. As we will discuss below, links need not be binary, though binary links are common. The term *item* refers to objects and links, and is denoted *e*.

An object or link contains a set A of *attributes*, where $|A|$x 0. In other work, attributes are called *variables* or *features* of an item. For example, attributes on objects might denote the age of an actor, the genre of a movie, or the location of a studio. Attributes on links might denote the salary an actor was paid for acting in a given movie or the role which an actor played in a movie. For a given item e_i, the *j*th attribute $a_{i,j}$ has a name *name*($a_{i,j}$) and a value *value*($a_{i,j}$). Items can be characterized by their attribute vector <*value*($a_{i,1}$), *value*($a_{i,2}$),...*value*($a_{i,n}$)>. For convenience, *name*(A_i) and *value*(A_i) represent sets of names and values, respectively, on the *i*th item. Attributes with identical names are assumed to have values of comparable data types. For example, all values of *age* are assumed to be integers, and all values of *release date* are assumed to be dates.

Items typically can be grouped into one or more disjoint subsets of homogeneous type. For example, objects in the domain of movies could be grouped into persons, movies, studios, and awards. We do not assume that there is only one "correct" typing system over

[1] The movie examples presented throughout this paper are drawn from our work with the Internet Movie Database (www.imdb.com). The IMDb is a large relational datatset that catalogues diverse information about movies, containing data for 300,000 movies, 700,000 people (e.g. actors, directors, producers), and 12,000 studios.

the objects and links represented in the data. Indeed, one system we discuss below implicitly attempt to learn a useful typing system. That said, databases often have an initial typing system, and items of a single type within that system often have equivalent attribute sets, $name(A_i) = name(A_j)$ for all i,j. Typing systems for items are often called an *ontology*, particularly when types are organized into a hierarchical structure that allows generalization about the lower-level entities (e.g., a director is a person and thus has birth date).

Several nearly equivalent formalisms exist for representing these types of data sets. For example, data sets could be represented as:

- *Graphs* — A data set D can be thought of as a directed hypergraph with vertices O and hyperedges L. The hypergraph may have only one connected component or several connected components.
- *Database tables* — A data set D can also be thought of as a relational database with entities O and relations L. Items of each type would be stored in a separate table with one field for each attribute.
- *First-order logic statements* — D can be thought of as a collection of statements in first-order logic.

For purposes of this paper, we use the formalisms of graph theory and use binary links, though no restriction to binary relations is implied.[2]

In contrast to the assumptions underlying of much of propositional learning, attribute vectors characterizing objects in D are not assumed to be statistically independent or identically distributed. For example, two movies made by the same director are likely to be of similar genres and made in similar years.

2.2 Models

Algorithms for probabilistic relational learning have been developed that learn probability distributions over possible attribute values, possible links, or possible objects. A model over possible attribute values might predict the box-office receipts of a movie based on the success of previous movies made by the movie's director, producer, and studio. A model over possible links might predict the probability that a given actor will star in a future movie, represented as an *acted-in* link between the movie object and that actor. A model over possible objects might predict the probability that a movie will be released in a particular country, represented as a new *release* object for that movie (with an appropriate link to the movie itself).

We focus on models that estimate probability distributions over possible attribute values, though much of our discussion is also relevant to prediction of links and objects. The task of estimating probability distributions over the values of a given attribute would appear to differ little from traditional propositional learning. However, algorithms for

[2] The apparent limitations imposed by binary links are largely illusory and can usually be escaped by creating objects to represent more complex relations such as events (see Davidson 1967).

relational learning typically look beyond the item for which the attribute is defined, to consider the effect of related objects on the probability distribution. For example, in order to predict the box-office success of a movie, a relational model would consider not only the attributes of the movie, but attributes of the actors in the movie and the director, producer, and studio that made the movie. A model might go even further and consider attributes of much more "distant" objects (in the sense of a graph neighborhood).

Some algorithms for relational learning estimate a probability distribution over the values of one or more attributes of item e_i given the items in its neighborhood. That is, $P(value(a_{i,j})|neighborhood(e_i))$, where $neighborhood(e_i)$ is a set of objects and links reachable from e_i and it is typically less than the entire graph. Others estimate the full joint probability distribution over a large number of attributes of related objects. Below, we discuss both types of models.

We divide the set of all attributes A into fixed and inferred attributes. For each fixed attribute A_j, value($a_{i,j}$) is known a priori for each item e_i. No inference about the probability distribution of a fixed attribute is necessary. In contrast, for each inferred attribute A_j, value($a_{i,j}$) is missing for one or more items e_i, and the goal of modeling is to estimate a probability distribution for these missing values. Friedman et al. call these "fixed" and "probabilistic" attributes, respectively.

2.3 Related work

Probabilistic relational learning is distinct from a set of related learning tasks that appear superficially similar:

- *Traditional inductive logic programming* — Traditional ILP systems learn deterministic models rather than estimating probability distributions over attributes, links, or objects. Recent developments in stochastic logic programming (Muggleton 2000, Cussens 2001) and Bayesian logic programs (Kersting and De Raedt 2000) are interesting alternatives to these deterministic models, which adapt ILP techniques to learn probabilistic first-order models.
- *Propositionalization of relational data* — A common approach to learning from relational data is to "propositionalize" the data rather than retain its inherent structure. The attributes of an item e_i can be "pulled back" and recorded as a local attribute of e_i. Then a probabilistic model can be learned from the resulting feature-vector representation of the data. However, this approach requires foreknowledge of the correct relational features. It also assumes that the nature of statistical inference is not changed by the relational structure of the data, an assumption that has been challenged by recent work showing how the structure of relational data affects parameter estimates (Jensen and Neville 2002a) and evaluation of learned models (Jensen and Neville 2002b).
- *Learning propositional concepts in relational databases* — Much of the work on knowledge discovery in databases addresses how to use techniques from relational databases to facilitate data mining. This work deals with relational databases, but

relatively little of it deals with relational data. Much of this work assumes that data are propositionalized (see above) or that data are drawn from only a single relation.

- *Multiple instance learning* — Some recent work has focused on how to learn in tasks where each data item is a "bag" containing multiple instances and where a positive bag indicates that one or more of the items it contains is positive. For example, a credit card account may be a bag of transactions, where some of the transactions in a given bag may be fraudulent. In relational learning, a data instance can be thought of as an item surrounded by a neighborhood of other items. However, in relational learning, the other items only provide context for estimating the probability distribution of the attributes of a single item, and do not represent multiple instances within a bag.

3 Example Systems

3.1 Probabilistic Relational Models

Probabilistic relational models (PRMs) extend Bayesian networks to support reasoning in relational domains, allowing the rich relational information to be incorporated into traditional Bayes net dependency structures. PRMs specify a probability model over a relational database with a fixed schema. Given a set of objects and the links between them, a PRM defines a full joint probability distribution over the attribute values of the objects. Attributes of an object can depend probabilistically on other attributes of the object, as well as on attributes of objects in its relational neighborhood. PRMs can also model uncertainty over both object and link existence.

PRMs make use of pre-specified database schemas that describe a fixed set of object types, each with a set of attributes and an associated set of links. A PRM represents a probability distribution over possible instances of a given schema, where an instance of the schema specifies (1) the set of objects of each type, (2) the set of links between the objects, and (3) the values of all attributes. For a particular skeleton set of objects and links (with missing attribute information), instantiating a PRM induces an unrolled Bayesian network model where random variables represent the individual attribute values of all the objects in the skeleton.

Instead of defining the dependency structure over attributes, as in conventional Bayes nets, PRMs define a generic dependency structure at the level of object *types*. Each attribute A_i associated with object type X is linked to a set of parents that influence the value of A_i. Parents of A_i are either (1) other attributes associated with type X, or (2) attributes associated with objects of type Y where objects Y are linked to objects X. (PRMs can also represent dependencies along longer chains of relations but for simplicity we will limit discussion to direct relations.) For the latter type of dependency, if the relation between X and Y is one-to-many, the “parent” consists of a set of attribute values. In this situation, PRMs use standard database aggregation functions (e.g. *max*, *mode*, *average*) to map sets of values into single values. Learning a PRM consists of two tasks: learning the dependency structure, and estimating the parameters of the conditional probability distributions used to specify the local probability models for an attribute given its parents.

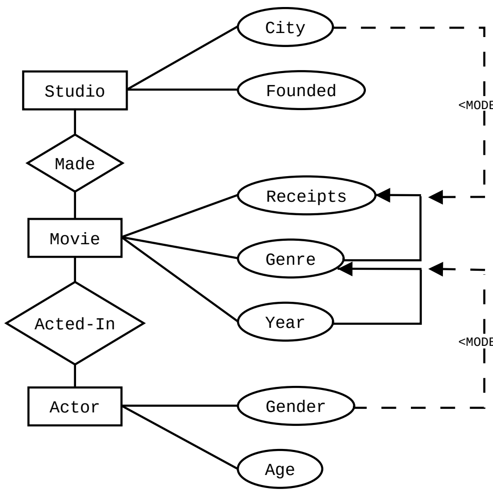


Fig. 1. The schema of a relational database in the movie domain and the dependencies among attributes encoded by a PRM.

For example, figure 1 shows an example of a PRM that could have be learned in the movie domain. Three types of objects in the schema are represented by boxes: studios, movies, and actors. Each type has a set of associated attributes represented by ovals. Links, represented by diamonds, relate studios and actors to movies. Directed arcs represent the learned dependency structure of a PRM. Dashed lines indicate an aggregated attribute – multiple actors may star in any particular movie and more than one studio may be involved in production of each movie. `<MODE>` annotations on the arcs indicate that the model uses the most prevalent attribute value from sets while reasoning. The model outlines the dependency of movie genre on both the year of the movie and the most prevalent gender of the actors in the movie. The model also shows that genre and most prevalent studio location influence movie success (box office receipts).

3.2 First-order Bayesian Classifiers

1BC is a simple Bayesian Classifier that builds discriminative models of attributes from relational data. Examples consist of objects and their relational neighborhood and first-order features evaluate attributes values of various items in the examples. A probability density function is specified for the target class using first-order features, making the standard naïve Bayes assumption that features are conditionally independent given the class. As with PRMs, it is assumed that the domain provides a well-defined notion of object types (individuals), with associated sets of links (structural predicates) and attributes (property predicates).

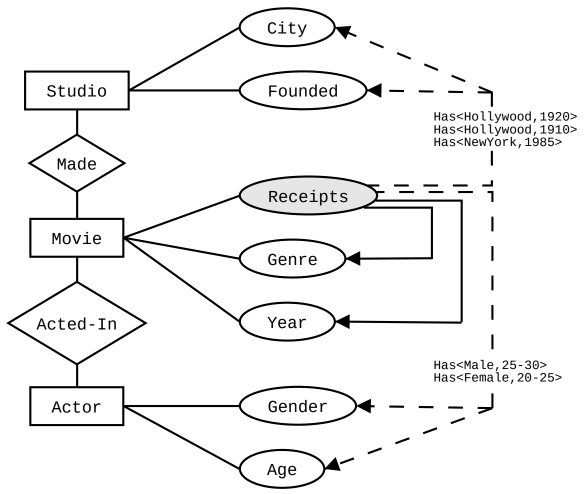


Fig. 2. The schema of a relational database in the movie domain and the dependencies among attributes encoded by a 1BC model.

1BC is designed to model at varying levels of decomposition. Level-0 does not decompose the examples at all and uses complex probability estimators over sets and multisets to model objects with heterogeneous relational neighborhoods in terms of their components. If there are too few examples to produce accurate probability estimates at this level, the examples can be decomposed into level-1 elementary features that consider the items of each example individually. For each original type *X* and its set of attributes *A*, new types are defined for combinations of attributes in *A*. Boolean features are constructed for these types which denote whether a particular example contains an item of the specified type. For example, the feature *HasActor<Female,20-25>* is true of a movie example is there exists a female actress in the age range (20,25) starring in the movie. A level-2 decomposition further subdivides these types by the attributes and constructs features that reference a single attribute of an item, such as *HasActor<Male>*.[3]

[3] Work by Craven and Slattery (2001) is related to 1BC, but it uses a complementary method. Rather than use an ILP system to construct features for a Bayesian classifier, Craven and Slattery use a Bayesian classifier to construct predicates for an ILP system.

The implementation of 1BC uses Tertius (Flach and Lachiche 2001) to return a set of interesting first-order features at each level of decomposition, given domain and background knowledge. These first-order features consist of zero or more links which reference specific items of an example and attributes of that item. Features are multi-valued if the links reference an attribute of a single item (e.g. movie) or boolean if the series of links refer to a set of items (e.g. actors).

Figure 2 shows an example a 1BC model that could be used to classify movie success. First-order features use the relational context to predict a movie's box office receipts – attributes of the movie itself are considered (e.g. genre, year) as well as attributes of related objects (e.g. studio city and founding year, actor age and gender). Each feature is used as an attribute in the naïve Bayes formula and the conditional probabilities of features given class are estimated from the training data.

3.3 Relational Probability Trees

Relational Probability Trees (RPTs) extend standard probability estimation trees to a relational setting. RPTs are used in PROXIMITY, a relational knowledge discovery system designed to operate on graph databases.[4] The RPT algorithm takes a collection of subgraphs as input and constructs a probability estimation tree to predict the target class label. Each subgraph in the collection contains one target object to be classified; the other items in the subgraph form its relational neighborhood. An RPT encodes a probability distribution over the class label given attributes of both the target objects and of other items in the subgraphs.

The subgraphs in the input collection specify the relational neighborhood that will be considered by the model and their structure defines a typing over items in the collection. Subgraphs are extracted from a larger graph database using the visual query language QGRAPH (Blau, Immerman, and Jensen 2001). Queries in the language allow for variation in the number and types of objects and links that form the subgraphs and return collections of all matching subgraphs from the database. Objects and links in the returned collection are named for particular roles that they serve in the matched subgraphs and may be duplicated across subgraphs if they match in multiple ways. The returned collection is a view of the graph — subgraph membership defines the relevant local relational context and names serve to dynamically assign object and link types. After querying, the user specifies a set of attributes for each type that will be available to the model during the learning phase.

The RPT induction algorithm searches over a space of relational features involving attributes of items in the input subgraphs. Relational features are similar to traditional propositional features in that they identify both an attribute and a way of testing the values of the attribute. However, relational features may also identify a particular relation (e.g. *ActedIn*) that links a single object (e.g. movie) to a set of other objects (e.g. actors). If this

[4] For additional details on PROXIMITY, see <http://kdl.cs.umass.edu>.

is the case, the attribute referenced by the feature may belong to the linked objects (e.g. actor age), and the test is conducted on the set of attribute values on the linked objects. The algorithm searches over a space of possible item types (e.g. actor), attributes (e.g. age), and aggregation functions (e.g. count). Each node in an RPT tests a binary relational feature — for example, whether at least two actors in a movie are over 65.

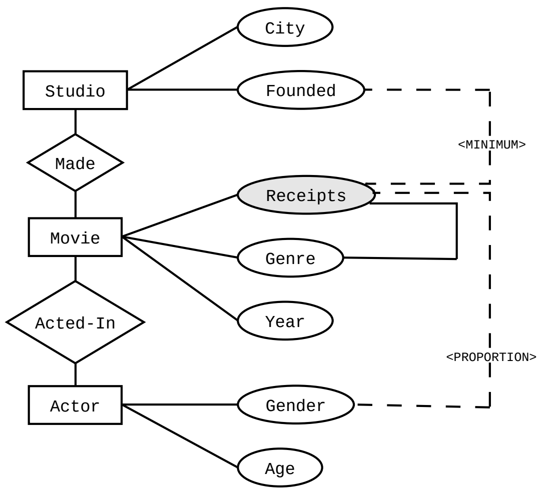


Fig. 3. The schema of a relational database in the movie domain and the dependencies among attributes encoded by a RPT.

Figure 3 shows the dependencies of an RPT for classifying movie success. The model has selected three attributes relevant to predicting a movie's box office receipts. Features of the model test attributes of the movie (e.g. genre) and attributes of related objects (e.g. studio founding year, actor gender). Annotations on the arcs indicate the aggregations used in the features. More specifically, the tree could test whether the earliest founding year of studios is below a threshold (e.g. <1910) and whether the proportion of actor genders satisfies some constraint (e.g. females<20%).

4 Schema-model Framework

Each of the learning techniques outlined above has roots in techniques for learning in propositional data. PRMs extend traditional Bayesian networks, first-order Bayesian classifiers extend traditional Bayesian classifiers, and RPTs extend techniques from probability estimation trees (Provost and Domingos 2000). However, each technique also requires substantial modification to learn effectively from relational data.

We group these modifications into changes relevant to a *schema* and those relevant to a *model*. The *schema* addresses how the technique structures relational data for the modeling algorithm. The *model* addresses alterations to the modeling technique that handle the more complex structure of relational data.

4.1 Schema

In contrast to the highly specified task of propositional learning, relational learning raises new questions. For example, compare learning a model of the probability distribution of a movie's box-office receipts from propositional and relational data, respectively. In propositional data, the learner is given a fixed set of attributes intrinsic to each movie. In relational data, how much of the relational neighborhood around a movie should influence the probability distribution of a movie's box-office receipts? In propositional data, each attribute is considered as a separate entity. In relational data, should the attributes of the movie's director, producer, and actors contribute independently to predicting box-office receipts, or should we consider some aggregate attributes of all people connected with the movie? We call the collective answers to these questions a *schema*.

The schema provides access to an appropriate neighborhood of items that influence the probability distribution of inferred attributes. Either this neighborhood should include all items that influence the inferred attributes, or the influence of those items should be carried through an appropriate inference chain (see below). Second, the schema groups items into equivalence classes that can be used by the model.

In PRMs, the schema is a provided by a combination of the dependency structure of the PRM itself, which provides a set of parents for each probabilistic attribute, and the *database schema*, which groups objects and links into mutually exclusive and collectively exhaustive types.[5] For the purposes of this discussion, two features of PRMs are particularly noteworthy. First, the typing system is given by the underlying database, and cannot be altered without changing the structure of the database.[6] Second, the probability distribution of a given attribute appears to depend only on a given set of parents, but the inference procedure employed by PRMs allows probabilistic attributes to depend on other probabilistic attributes whose values are inferred by the model. Such an *inference chain* allows the inferred distribution of an attribute to be influenced by attributes beyond its immediate parents.

In 1BC, the schema is provided by a combination of the structure of the data graph, which is assumed to consist of disjoint subgraphs representing individual instances, and

[5] Note that the database schema is related to, but different than the schema for the learning technique.

[6] Other work related to PRMs has dealt with inducing relevant types from relational data (Taskar, Segal, and Koller 2001)

first-order features learned by an ILP system. Alternatively, the instance subgraphs could be formed by some sort of query mechanism, rather than relying on natural divisions of the data, although this option is not discussed in the literature on 1BC. Note that the first-order features are not learned with direct reference to their utility to the probabilistic model, though they are learned rather than being provided *a priori*.

In RPTs, the schema is provided by the results of a graph query executed prior to learning the model. The query returns a collection of subgraphs, each of which contains one target item with the attribute to be modeled. The query also associates names with items in each subgraph, which can be interpreted as types (e.g., movie, actor, producer, director, and studio). This approach to providing a schema sits midway between PRMs and 1BC; the schema is not fixed, but neither is it learned.

4.2 Model

A schema provides substantial additional structure to relational learning tasks, but creating a probabilistic model in the presence of a schema still raises unique questions. Again, compare learning a probability distribution of a movie's box-office receipts in the propositional and relational cases. In propositional data, every instance has an identical and fixed structure. In relational data, how should we deal with the varying size of the neighborhood surrounding a given movie? A model of relational data must accurately estimate probability distributions in the face of this varying structure.

PRMs address this challenge by aggregating over the attribute values of multiple parents of the same type. For example, if a movie's genre depends on the gender of the movie's actors, and a movie has five actors, then the PRM would aggregate the five values of gender with a function such as *mode*. These aggregation functions appear to be selected *a priori*, rather than selected by the learning algorithm itself.

1BC addresses the challenge of variable-sized neighborhoods by using first-order features that produce boolean values that characterize the neighborhood. For example, a first-order feature might determine whether an actor with a particular gender and age acted in the movie. The attribute would be true if one or more actors met the gender and age criteria. The first-order features are not formed during model learning. Instead, they are formed prior to learning the Bayes classifier.

The RPT induction algorithm also forms first-order features, but it forms these features as part of the same process that constructs the tree. The RPT induction algorithm searches the a space of possible types, attributes, aggregation functions, and thresholds to form binary predicates such as *average actor age < 30*. Thus, RPT induction creates similar types of first-order features as 1BC. RPT induction searches over a larger space of aggregation functions and thresholds than either PRMs or 1BC, but RPT induction lacks the ability of 1BC to automatically create long first-order chains of conditions (e.g., whether an actor over 30 also starred in an action adventure movie).

5. New problems in relational KD

5.1 Learning schemas

The quality of any probabilistic model of relational data depends on the features available to the modeling technique. In the language of schemas and models, model quality depends on schema quality. Ideally, the schema should be learned such that it provides the necessary "raw materials" for constructing an accurate probabilistic model.

First, the schema should provide access to an appropriate neighborhood of items that influence the probability distribution of inferred attributes. If the neighborhood is too small, and the influence of other relevant items is not carried through an inference chain, we conjecture that the model will have increased error due to *bias* (Friedman 1997) because it cannot represent the influence of some relevant items. If the neighborhood is too large, we conjecture that the model will have increased errors due to *variance* (Friedman 1997), because the data used for probability estimates will include irrelevant items that merely introduce "noise" that masks the "signal" provided by relevant items.

Second, the schema groups items into equivalence classes that can be used by the model to form relational features. We conjecture that this component of a schema must balance two types of variance errors. A schema might group items at a level of detail that is either too fine or too course. For example, in the movie domain, a schema might designate actors, directors, and producers as different types of objects, when the aggregate level of experience of all persons associated with a movie is the best predictor of the box-office success of a movie. Alternatively, a schema could designate actors, directors, and producers as "people" when the experience of directors and producers alone are the best predictors of success.

None of the techniques discussed above provides an entirely satisfactory solution to determining the correct schema. PRMs take the schema as given, based on the structure of the database. 1BC learns a set of relational features, but without regard to the effect of those features on the quality of the probabilistic model learned. RPTs assume a pre-specified schema, allowing alternative queries to be used, but not supporting any automated exploration of the space of possible schemas.

5.2 Alternative schemas

The relatively simple schemas used by current approaches do not exhaust the potential types of schemas that might be useful for learning. For example, PRMs and RPTs currently assume a single set of mutually exclusive and collectively exhaustive types. Alternatives to this approach include: 1) *probabilistic schemas*, where each item is characterized by a probability distribution over several possible types; 2) *overlapping schemas*, where each item can have multiple types; and 3) *multi-resolution schemas*, where each item is in a set of types of increasing generality. Further, it is not clear that one schema is appropriate for learning all probability distributions. This is particularly relevant to PRMs, which might benefit from learning different conditional probability distributions with different schemas.

5.3 Learning models

Finally, a variety of challenges remain in learning accurate probabilistic models from relational data. In related work (Jensen and Neville 2002a, 2002b), we have shown how common characteristics of relational data can complicate learning and evaluating relational models. Specifically, high levels of linkage and autocorrelation in relational data can cause learning algorithms to systematically prefer attributes with the *least* supporting evidence. Similarly, these characteristics of relational data can cause evaluations of learned models to vastly overestimate the utility of those models. In addition to these, we suspect that other statistical challenges will emerge in the next few years, as we gain experience with learning probabilistic models from relational data.

6 Related and Future work

Related work on Bayesian logic programs (BLPs) (Kersting and De Raedt 2000) has characterized the expressivity of several probabilistic models, including Probabilistic relational models (PRMs) (Freidman et. al 1999), relational Bayesian nets (RBNs) (Jaeger 1997), and probabilistic logic programs (PLPs) (Ngo and Haddawy 1997). Kersting and De Raedt compared these three types of models with BLPs and investigated the relationships among the various knowledge representations. They outlined a chain of positive inclusion for the four types of models, from PRMs (least expressive) to RBNs, to PLPs, to BLPs (most expressive). Instead of focusing on model representation, the SMF relates systems based on aspects of model learning. At the outset of our work on the SMF, PRMs were the only type of model considered by Kersting and De Raedt for which learning had been investigated. Consequently, we limit the initial discussion in this paper to PRMs. However, recent work on Bayesian logic programs (Kersting and De Raedt 2002) has outlined methods to learn both the parameters and structure. In future work, we hope to incorporate BLPs into the SMF.

For simplicity, our work was limited to systems that model degrees of belief over individuals (Halpern 1990). Halpern outlines two types of probabilistic structures in his analysis of first order logics of probability. The first type of structures represent subjective probabilities concerning particular individuals (e.g. any given bird is likely but not certain to fly), modeling probabilities over possible worlds. The second type of structures represent qualitative statistical statements (e.g. most birds fly), modeling probabilities over the domain. The systems described in this paper are, in Halpern's notation, Type I models. However, there is no reason for the schema-model framework to be limited to

Type I models. Work in Type II probabilistic models includes stochastic logics programs (Muggleton 2000, Cussens 2001) and PRISMs (Sato and Kameya 2001). In future work, we hope to incorporate these systems into the SMF.

Acknowledgment

This effort is supported by DARPA and AFRL under contract numbers F30602-00-2-0597 and F30602-01-2-0566. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements either expressed or implied, of the DARPA, the AFRL, or the U.S. Government.

References

1. Blau, H., N. Immerman, and D. Jensen. (2001). A Visual Query Language for Relational Knowledge Discovery. University of Massachusetts Amherst, Department of Computer Science. Technical Report 01-28.
2. Craven, M. and S. Slattery (2001). Relational learning with statistical predicate invention: Better models for hypertext. *Machine Learning* 43: 97-119.
3. Cussens, J. (2001). Parameter estimation in stochastic logic programs. *Machine Learning* 43:245-271.
4. Davidson, D. (1967). The logical form of action sentences. In *The Logic of Decision and Action*. N. Rescher (Ed.). University of Pittsburgh Press.
5. Flach, P. and N. Lachiche (1999). 1BC: A first-order Bayesian classifier. In *ILP'99*. 93-103.
6. Flach, P. and N. Lachiche (2001). Confirmation-guided discovery of first-order rules with Tertius. *Machine Learning* 42:61-95.
7. Friedman, N., L. Getoor, D. Koller and A. Pfeffer. (1999). Learning probabilistic relational models. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*. 1300-1309.
8. Friedman, J. (1997). On bias, variance, 0/1-loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery* 1:55-77.
9. Halpern, J. Y. (1990). An analysis of first-order logics of probability. *Artificial Intelligence* 46:311-350.
10. Jaeger, M. (1997). Relational Bayesian networks. In *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence*. 266-273.
11. Jensen, D. and J. Neville (2002a). Linkage and autocorrelation cause feature selection bias in relational learning. To appear in *ICML2002*.
12. Jensen, D. and J. Neville (2002b). Linkage and autocorrelation cause bias in evaluation of relational learners. To appear in *ILP2002*.
13. Kersting, K. and L. De Raedt. (2000). Bayesian logic programs. In *Proceedings of the Work-in-Progress Track at the 10th International Conference on Inductive Logic Programming* 138-155.

14. Kersting, K. and L. De Raedt. (2002). Basic principles of learning Bayesian logic programs. Technical Report No. 174, Institute for Computer Science, University of Freiburg, Germany.
15. Muggleton, S. (2000). Learning stochastic logic programs. In *Proceedings of the AAAI-2000 Workshop on Learning Statistical Models from Relational Data.* AAAI Press.
16. Neville, J. and D. Jensen (2000). Iterative classification in relational data. In *Proceedings of the AAAI-2000 Workshop on Learning Statistical Models from Relational Data.* AAAI Press. 13-20.
17. Neville, J. and D. Jensen (2002). Learning relational probability trees. University of Massachusetts Amherst, Department of Computer Science, Technical report.
18. Ngo, L. and P. Haddawy (1997). Answering queries from context-sensitive probabilistic knowledge bases. *Theoretical Computer Science* 171:147-177.
19. Provost, F. and P. Domingos (2000). "Well-trained PETs: Improving probability estimation trees." CeDER Working Paper #IS-00-04. Stern School of Business, New York University.
20. Sato, T. and Y. Kameya. (2001). Parameter learning of logic programs for symbolic-statistical modeling. *Journal of Artificial Intelligence Research* 15:391-454.
21. Taskar, B., E. Segal, and D. Koller (2001). Probabilistic clustering in relational data. In *AAAI-2001*. 870-876.

D. Jensen and J. Neville (2002). Linkage and autocorrelation cause feature selection bias in relational learning. Proceedings of the Nineteenth International Conference on Machine Learning (ICML2002). Morgan Kaufmann. pp. 259-266

Linkage and Autocorrelation Cause Feature Selection Bias in Relational Learning

David Jensen Jennifer Neville -- JENSEN@CS.UMASS.EDU JNEVILLE@CS.UMASS.EDU
Knowledge Discovery Laboratory, Computer Science Department, Univ of Mass, Amherst, MA 01003

Abstract

Two common characteristics of relational data sets — concentrated linkage and relational auto-correlation — can cause learning algorithms to be strongly biased toward certain features, irrespective of their predictive power. We identifythese characteristics, define quantitative measures of their severity, and explain how they produce this bias. We show how linkage and auto-correlation affect a representative algorithm for feature selection by applying the algorithm to synthetic data and to data drawn from the Internet Movie Database.

1. Introduction

Recent efforts to learn statistical models from relational data include work on stochastic logic programming (Muggleton 2000), probabilistic relational models (Getoor et al. 1999), and relational Bayesian classifiers (Flach and Lachiche 1999). Relational data representations greatlyexpand the range and applicability of machine learning,but the greater expressive power of relational representations produces new statistical challenges. Work on relational learning often diverges sharply from traditional learning algorithms that assume data instances are statistically independent. Statistical independence of instances is among the most enduring and deeply buried assumptions of traditional machine learning methods, and it is contradicted by many relational data sets.

In this paper, we focus on how dependence among the values of a class label in relational data can complicate feature selection in methods for machine learning. We define relational feature selection and give a simple example of how such a procedure can be biased. We define quantitative measures of concentrated linkage (L) and relational autocorrelation (C'), two common characteristics of relational data sets. We show how high values of L and C' reduce the effective sample size of some data sets,introduce additional variance, and lead to feature selection bias. To our knowledge, no current relational learning algorithm accounts for this bias. We show how to estimate the variance of scores and discuss using those estimates to improve feature selection in relational data.

1.1 Relational Data and Statistical Dependence

Figure 1 presents two simple relational data sets. In each set, instances for learning consist of subgraphs containing a unique object x, an object y, and one or more other objects. Objects x contain a class label and objects y contain an attribute that will be used to predict the class label of x. Figure 1a shows a data set where objects x and y have a one-to-one relationship and where the class labels on instances are independent. Figure 1b shows instances where objects x and y have a many-to-one relationship and where the class labels are dependent.[1]

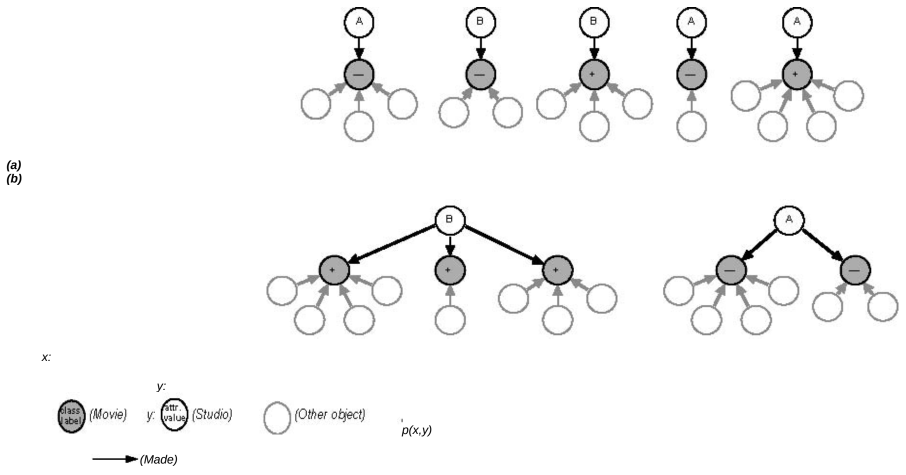


Figure 1: Example relational data sets with independent instances (a) and dependent instances (b).

We will spend the majority of the paper considering data sets similar in structure to Figure 1b where each subgraph consists of multiple relations and each relation may produce dependencies among the instances. For simplicity, our experiments assume that all relations are binary, placing this work somewhere between multiple instance learning and full first-order logic (DeRaedt 1998), although the statistical effects we investigate appear to affect a wider range of relational learning algorithms.

[1] Throughout this paper, we assume that all linkages among instances that could introduce statistical dependence are represented explicitly as links (edges) in the data graph.

1.2 Relational Feature Selection

This paper focuses on *feature selection*, a component of many learning algorithms. We define *feature* as a mapping between raw data and a low-level inference. For example, a feature for a propositional data set about medical patients might be *temperature > 99°F.* In this case, a feature combines an attribute (*temperature)*, an operator, and a value. Typically, many features are combined into a higher-level model such as a tree or rule set. We define *feature selection* as any process that chooses among features, either by identifying the best, selecting some and rejecting others, or merely placing a partial or total ordering over all possible features. This definition is broader than some (e.g., John, Kohavi, and Pfleger 1994), but it emphasizes the central role of feature selection in machine learning algorithms, including algorithms for learning decision trees, classification and association rules, and Bayesian nets. Feature selection is central to any learning algorithm that forms models containing a subset of all possible features.

We focus here on *relational* feature selection. Relational features are used by models that predict the value of an attribute on particular types of objects (e.g., the box office receipts of movies) based on attributes of related objects (e.g., characteristics of the movie's director, producer, actors, and studio). Relational features are similar to the features described above in that they identify both an attribute and a way of testing the values of the attribute. However, relational features may also identify a particular relation (e.g. *ActedIn(x,y))* that links a single object *x* (e.g. movie) to a set of other objects *Y* (e.g. actors). If this is the case, the attribute referenced by the feature may belong to the related objects *Y* (e.g. age), and the test is conducted on the set of attribute values on the objects in *Y*. For example, the relational feature:

$$Max(Age(Y)) > 65 \quad \text{where} \quad Movie(x), \quad Y = \{y \mid ActedIn(x,y)\}$$

determines whether the oldest of the actors in movie *x* is over 65. Throughout this paper, we will use *f(x)* to refer the value of attribute *f* for a single object *x*, and *f(X)* to refer to the set of values of attribute *f* for all $x \in X$.

A central characteristic of many relational data sets is that two or more objects of one type (e.g., movies) can both be connected to the same object of another type (e.g., a studio). We expect that this shared linkage represents some statistical associations present in the world. That is, linked objects are not statistically independent.

By examining relational feature selection in general, this work is relevant to nearly any learning algorithm that compares and selects among different relational features, including algorithms for inductive logic programming (Dzeroski & Lavrac 2001) and algorithms for constructing relational versions of commonly used model representations such as rules, trees, and Bayesian networks (Getoor et al. 1999).

1.3 An Example: Bias in Relational Feature Selection

Given that instances in relational data may not be independent, we should examine how such relational structure could affect feature selection. Below we show how relational structure and dependence among values of the class label can bias relational feature selection. To do this, we created data sets about movies and analyzed them with a simple algorithm for relational feature selection. Specifically, we created and analyzed a family of relational data sets whose relational structure was drawn from the Internet Movie Database (www.imdb.com). We gathered a sample of 1383 movies released in the United States between 1995 and 2000. In addition to movies, the data set contained objects representing actors, directors, producers, and studios. The data schema is shown in Figure 2.

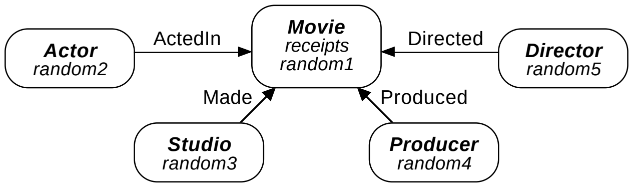


Figure 2: Schema for the movie data sets. Object and link types are shown in roman; attributes are shown in italics.

We created a learning task using a single attribute on movies — opening-weekend box office receipts. We discretized that attribute so that a positive value indicates a movie with more than \$2 million in opening weekend receipts (*prob(+)=0.55*). We call this discretized attribute *receipts* and use it as a binary class label. For each of 1000 trials, we also generated a random binary attribute on each of the five types of objects (movies, studios, actors, directors, and producers).

In each trial, we applied an algorithm for relational feature selection, using features formed from the random attributes. The algorithm uses the following relational feature for each attribute *f(x)* and each attribute value a_f:

$$Mode(f(Y)) = a_f$$
$$\text{where} \quad Movie(x), \quad Y = \{y \mid LinkedTo(x,y) \wedge Type(y) = t\}$$

which determines whether the modal value of *f* on the objects of type *t* linked to *x* is equal to a_f.

The correlation of each relational feature with the class label was calculated using chi-square and the features were ranked by their chi-square values. In each trial, we identified the object type of the top-ranked feature.

Given that all attributes were created randomly, we would expect an algorithm to select all features with equal probability, since no attribute is useful for predicting *receipts*. However, as shown in the first column of Table 1, features formed from studio objects have a much higher probability of selection than features formed from movies, actors, directors, or producers.

Table 1: Probability of feature selection

Object Type	*Receipts* Class Label	Random Class Label
Studio	0.742	0.214
Director	0.059	0.218
Producer	0.073	0.174
Actor	0.072	0.186
Movie	0.054	0.208

This effect is eliminated if the values of *receipts* are assigned randomly (with the same probability as before) instead of using the actual values of *receipts*. These results are shown in the second column of Table 1. For a random class label, the algorithm has no bias toward studio objects. It behaves in the way we would expect, selecting among features formed from different object types with roughly equal probability. This raises obvious and intriguing questions: Why does the algorithm prefer random features formed from studios, and what does this tell us about relational feature selection in general?

2. Linkage and Autocorrelation

Our analysis indicates that bias such as that shown in Table 1 occurs when algorithms ignore two common characteristics of relational data — *concentrated linkage* and *relational autocorrelation*. We define these characteristics formally below. Informally, concentrated linkage occurs when many objects are linked to a common neighbor, and relational autocorrelation occurs when the values of a given attribute are highly uniform among objects that share a common neighbor. The example in Figure 1b shows several movies linked to an individual studio and shows that movies made by the same studio have highly correlated class labels.

2.1 Concentrated Linkage

We will define concentrated linkage *L(X,P,Y)* with respect to two sets of objects *X* and *Y* and a set of paths *P* such that the relation *p(x,y)* holds. Paths are composed of *k* links and *k-1* intervening objects, where *k≥1*. Each path represents a series of relations linking an object in *X* to an object in *Y*. For example consider the path from linking two movies, m_1 and m_2, made by the same studio. The path is formed from two *Made* links, *Made*(m_1,s_1) and *Made*(m_2,s_1). For convenience we treat all links as undirected in order to refer to meaningful sequences of relationships as paths. We assume that paths in *P* are unique with respect to a given *(x,y)* pair; if two or more paths between *x* and *y* exist in the data, they are collapsed to a single element of *P*.

Definition: D_{yX} is the degree of the object *y* with respect to a set of objects *X*. That is, the number of *x*∈*X* such that *p(x,y)*∈*P*. For example, D_{yX} might measure, for a given studio *y*, the number of movies (*X*) it has made. ■

Definition: *Single linkage* of *X* with respect to *Y* occurs in a data set whenever, for all *x*∈*X* and *y*∈*Y*:

$$D_{xY} = (\quad and \quad D_{yX} \geq (\qquad ■$$

In these cases, many objects in *X* (e.g., movies) connect to a single object in *Y* (e.g., a studio). We use single linkage as an important special case in future discussions.

Definition: The *concentrated linkage L(x,X,P,Y) of an individual object x* (e.g., a movie) that is linked to objects *Y* (studios) via paths *P* is:

$$L(x,X,P,Y) = \sum_{\substack{y\ s.t.\\ p(x,y)\in P}} \frac{(D_{yX}-1)}{D_{yX}} \Big/ {D_{xY}}^2 \qquad ■$$

the quantity $(D_{yX}\ -1)/D_{yX}$ within the summation is zero when the D_{yX} is one, and asymptotically approaches one as degree grows, and thus is a reasonable indicator of *L(x,X,P,Y)*, given single linkage of *x* with respect to *Y*. Because *x* may be linked to multiple nodes in *Y*, we define the average across all nodes y_i linked to *x*, and divide by an additional factor of D_{xY} to rate single linkage more highly than multiple linkage.

Definition: The *concentrated linkage L(X,P,Y) of a set of objects X* (e.g., all movies) that are linked to objects *Y* is:

$$L(X,P,Y) = \sum_{x\in X} \frac{L(x,X,P,Y)}{|X|} \qquad ■$$

Given particular types of linkage, *L* can be calculated analytically from the sufficient statistics |*X*| and |*Y*|. For example, in the case of single linkage of *X* with respect to *Y*, *L* = (|*X*|-|*Y*|)/|*X*|. For example, the data set shown in Figure 1b exhibits single linkage, so *L(X,P,Y)* = 0.60. Propositional data also display single linkage, and because |*X*|=|*Y*|, *L(X,P,Y) = 0*. Calculations of several types of linkage are shown for the movie data in Table 2.

Table 2: Linkage in the movie data

Linkage Type	Value
L(Movie, Made, Studio)	0.91
L(Movie, Directed, Director)	0.23
L(Movie, Produced, Producer)	0.08
L(Movie, ActedIn, Actor)	0.01

In addition to the movie data, we have encountered many other instances of concentrated linkage. For example, while studying relationships among publicly traded companies in the banking and chemical industries, we found that nearly every company in both industries uses one of only seven different accounting firms. In work on fraud in mobile phone networks, we found that 800 numbers, 900 numbers, and some public numbers (e.g., 911) produced concentrated linkage among phones. Concentrated linkage is also common in other widely accessible relational data sets. For example, many articles in the scientific literature are published in a single journal and many basic research articles are cited in single review articles. On the Web, many content pages are linked to single directory pages on sites such as Yahoo.

2.2 Correlation and Autocorrelation

We will define relational correlation *C(X,f,P,Y,g)* with respect to two sets of objects *X* and *Y*, two attributes *f* and *g* on objects in *X* and *Y*, respectively, and a set of paths *P* that connect objects in *X* and *Y*.

Definition: *Relational correlation C(X,f,P,Y,g)* is the correlation between all pairs *(f(x),g(y))* where $x \in X$, $y \in Y$ and $p(x,y) \in P$. ■

Given the pairs of values that these elements define, traditional measures such as information gain, chi-square, and Pearson's contingency coefficient can be used to assess the correlation between values of the attributes *f* and *g* on objects connected by paths in *P*. The range of *C* depends on the measure of correlation used.

We can use the definition of relational correlation *C(X,f,P,Y,g)* to define relational *autocorrelation* as the correlation between the same attribute on distinct objects belonging to the same set.

Definition: *Relational autocorrelation C' is:*

$$C'(X,f,P) \equiv C(X,f,P,X,f) \quad \text{where} \quad \forall p(x_i,x_j) \in P \quad x_i \neq x_j \blacksquare$$

For example, *C'* could be defined with respect to movie objects, the attribute *receipts* on movies, and paths formed by traversing *Made* links that connect the movies to an intervening studio.

If the underlying measure of correlation varies between zero and one, then *C'=1* indicates that the value of the attribute for a specific node x_i is always equal to all other nodes x_j reachable by a path in *P*. When *C'=0*, values of *f(X)* are independent. Table 3 gives estimates of relational autocorrelation for movie receipts, linked through studios, directors, producers, and actors. For a measure of correlation, Table 3 uses Pearson's corrected contingency coefficient (Sachs 1992), a measure that produces an easily interpreted value between zero and one. Autocorrelation is fairly strong for all object types except actors.

In addition to the movie data, we have encountered many other examples of high relational autocorrelation. For example, in our study of publicly traded companies, we found that when persons served as officers or directors of multiple companies, the companies were often in the same industry. Similarly, in biological data on protein interactions we analyzed for the 2001 ACM SIGKDD Cup Competition, the proteins located in the same place in a cell (e.g., mitochondria or cell wall) had highly autocorrelated functions (e.g., transcription or cell growth). Such autocorrelation has been identified in other domains as well. For example, fraud in mobile phone networks has been found to be highly autocorrelated (Cortes et al. 2001). The topics of authoritative web pages are highly autocorrelated when linked through directory pages that serve as "hubs" (Kleinberg 1999). Similarly, the topics of articles in the scientific literature are often highly autocorrelated when linked through review articles.

Table 3: Autocorrelation in the movie data

Autocorrelation Type	Value
C'(Movie,Receipts,Made|Studio|Made)	0.47
C'(Movie,Receipts,Directed|Director|Directed)	0.65
C'(Movie,Receipts,Produced|Producer|Produced)	0.41
C'(Movie,Receipts,ActedIn|Actor|ActedIn)	0.17

Note: We use a the notation *a*|*x*|*b* to denote paths with links of type *a* and *b* and intervening objects of type *x*.

We have defined relational autocorrelation in a similar way to existing definitions of temporal and spatial autocorrelation (see, for example, Cressie 1993). Autocorrelation in these specialized types of relational data has long been recognized as a source of increased variance. However, the more general types of relational data commonly analyzed by relational learning algorithms pose even more severe challenges because the amount of linkage can be far higher than in temporal or spatial data and because that linkage can vary dramatically among objects.

Relational autocorrelation represents an extremely important type of knowledge about relational data, one that is just beginning to be explored and exploited for learning statistical models of relational data (Neville and Jensen 2000; Slattery and Mitchell 2000). Deterministic models representing the extreme form of relational autocorrelation have been learned for years by ILP systems. By representing and using relational autocorrelation, statistical models can make use of both partially labeled data sets and high-confidence inferences about the class labels of some nodes to increase the confidence with which inferences can be made about nearby nodes.

However, as we show below, relational autocorrelation can also greatly complicate learning of all types of relational models. As we seek to represent and use relational autocorrelation in statistical models of relational data, we will need to adjust for its effects when evaluating more traditional types of features in these models.

2.3 Discussion

The results reported so far for concentrated linkage and relational autocorrelation provide important clues to the behavior reported in Table 1. Figure 3 plots all objects in the movie data in terms of their linkage and autocorrelation with respect to movies, as reported in Tables 2 and 3. The contours in the plot delineate regions where the severity of the bias introduced by linkage and autocorrelation is approximately equal. The contours are a 2-D view of the results reported in Figure 4 (described in detail in section 3.1). Studios objects in the movie data have the highest combination of concentrated linkage and relational autocorrelation. Features that use studios also show the greatest bias in the experiments reported in Table 1. While directors have a higher value of autocorrelation C', their linkage L is quite low. As we will show in the next section, when linkage and autocorrelation are both high for a single type of object, they bias learning algorithms toward features formed from objects of that type.

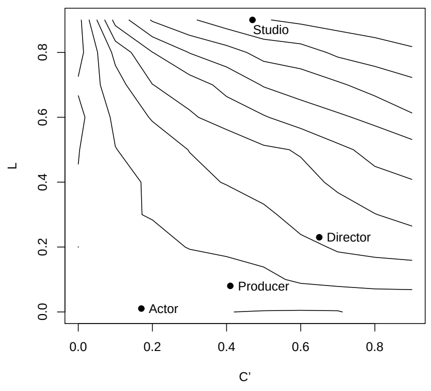


Figure 3: Relational autocorrelation vs. concentrated linkage

3. Effects of Linkage and Autocorrelation

Linkage and autocorrelation cause feature selection bias in a two-step chain of causality. First, linkage and autocorrelation combine to reduce the *effective sample size* of a data set, thus increasing the variance of scores estimated using that set. Relational data sets with high linkage and autocorrelation contain less information than an equivalently sized set of independent data. This reduction in effective sample size increases the variance of parameter estimates made with the data. Just as small data samples can lead to inaccurate estimates of the scores used to select features, concentrated linkage and autocorrelation can cause the scores of some features to have high variance. Second, increased variance of score distributions increases the probability that features formed from objects with high linkage and autocorrelation will be selected as the best feature, even when these features are random.

3.1 Decreased Effective Sample Size

Below, we prove a special case of linkage and autocorrelation decreasing effective sample size, for data exhibiting single linkage and in which $C'=1$ and $L \geq 0$. Then we explore a wider array of values for C' and L via simulation. Specifically, in data sets exhibiting single linkage, and where $L \geq 0$ and $C'=1$, the variance of scores estimated from relational features depends on $|Y|$ (the number of objects with an attribute value) rather than $|X|$ (the number of objects with a class label). For example, in the movie data, if autocorrelation of movie *receipts* through studios were perfect ($C'=1$), then the variance of scores for predicting *receipts* with a relational feature formed from studios (e.g., location) would depend on the number of studios in the sample rather than the number of movies.

Theorem: Given a relational data sample with objects X, objects Y, paths P, a class label $f(x)$, and an attribute $g(y)$, where $C'=1$, $D_{xY}=1$, and $D_{yX} \geq 1$, the sampling distribution for the scoring function $S(f,g)$ will have the same variance as it would for a data set with $|Y|$ independent instances.

Proof sketch: Consider the set of independent instances shown in Figure 1a, where $L=0$ (and, thus, $|X| = |Y|$). If we alter the data set so that $L>0$ (and, thus, $|Y|<|X|$), but retain the constraints that $C'=1$, $D_{xY}=1$, and $D_{yX} \geq 1$, then additional objects X will be added, and their corresponding values of $f(x)$ will match the value of other objects X already connected to a given Y. Such alterations increase the number of objects $|X|$, but they do not alter the number of possible arrangements of values of f and g. That number remains the same, because the value of $f(x)$ for additional objects X linked to a given Y is completely determined by the existing value of $f(x)$, given that $C'=1$. Each sample for which $L>0$, $C'=1$, and $D_{xY}=1$ corresponds directly to a sample for which $L=0$ though the latter sample contains fewer objects X. The number of ways of assigning values of f and g to objects is identical, the probability of each of these corresponding data sets remains equal, and the sampling distribution of any scoring function will also be identical. ■

In the independent case, the effective sample size $N = |X| = |Y|$. In the case where $L>0$, the effective sample size $N = |Y|<|X|$. In less extreme cases, where $0<C'<1$, the effective sample size lies somewhere between $|X|$ and $|Y|$. In addition, forms of linkage where $D_{xY}>1$ complicate estimates in ways we do not consider formally in this paper, although our experimental results below give some indications of the effect.

Simulation can demonstrate the effect of varying values of C' and L. We generated data sets with 1000 objects X with varying degrees of concentrated linkage to, and relational autocorrelation through, another set of objects Y ($|Y|$ varies between 1000 (L=0) and 100 (L=0.9)). We generated a binary class label on X and a binary attribute on Y, both drawn from a uniform distribution. At each level of linkage and autocorrelation, we generated 10,000 sets, calculated the chi-square score between the attribute and class label for each set, and then estimated the 95% confidence threshold for the resulting distribution. Because chi-square increases proportionally with sample size for a given level of association, we can find the effective sample size by dividing the 95% critical value of chi-square for independent data (3.84) by the 95th percentile of the distribution of simulated scores and multiply by the sample size (1000). The results are summarized in Figure 4.

In Figure 4, effective sample size drops monotonically with increases in C' and L. At extreme values of linkage and autocorrelation, effective sample size is reduced by almost an order of magnitude.

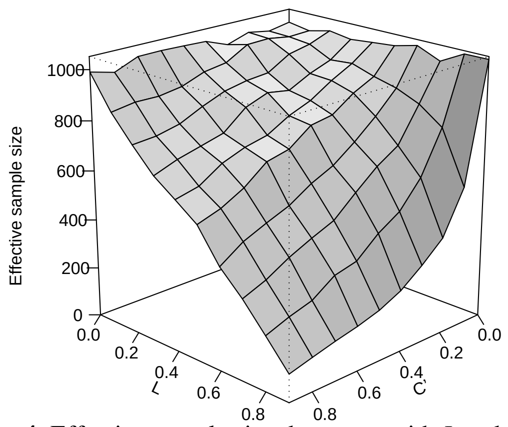


Figure 4: Effective sample size decreases with *L* and *C*'

We used similar means to gauge the effects of linkage and autocorrelation on the movie data, and estimated effective sample sizes for each object type. We examined the distribution of 200 random relational features formed using each object type. Autocorrelation and linkage should have no effect on movies (because movies are not linked directly to other movies), and the score distributions for features formed from movies approximately match the distribution of chi-square expected under the assumption of independent instances. The other distributions, however, differ substantially from this expectation. Table 4 shows the effective sample sizes obtained by minimizing the sum of the absolute difference at all percentiles of the empirical and theoretical chi-square distributions (other measures of similarity produced similar results).[2] In all cases, the assumed sample size would be equal to, or larger than, the number of movies (1383).

Table 4: Effective sample size in the movie data

Object Type	Scaling Factor	Effective Sample Size
Studio	0.026	36
Director	0.842	1164
Producer	0.615	851
Actor	0.702	971
Movie	1.000	1383

Figure 5 shows the distributions of scores obtained from testing the relational features used to construct Table 1. The dotted line shows the score distribution for studios. The solid lines show the overlapping distributions for movies, actors, directors, and producers. These latter distributions have quite similar variance, but the variance for features on studios is much higher. For these experiments, we used a chi-square statistic augmented with a sign, to indicate which diagonal of the contingency table contained the most mass (see Sachs 1992). In this way, we obtained a symmetric scoring function and we were able to tell if the utility of a given feature changed sign between data sets.

Evidence in Table 1, Figure 3, Table 4 and Figure 5, all points to common conclusions. Studios have the highest combination of linkage and autocorrelation, and the distributions of scores for features formed from studios display the highest variance. This variance reduces the effective sample size, and causes feature selection algorithms to be biased in favor of these features.

3.2 Feature Selection Bias

Given that the scores of some features are estimated with higher variance than others, why should this lead to a bias in favor of these attributes? Recent work on the statistical effects of multiple comparison procedures on score distributions (Jensen & Cohen 2000) provides an explanation.

Features are usually formed by a local search over possible parameters of the feature. For example, forming the feature mentioned earlier — *Max(Age(Y)) > 65* — could involve local search over many possible aggregation functions (*Max, Min, Average*), operators (*>, <, =*), and values (*[0,100]*). This local search is usually done prior to feature selection, so only the best feature from each feature "family" is compared.

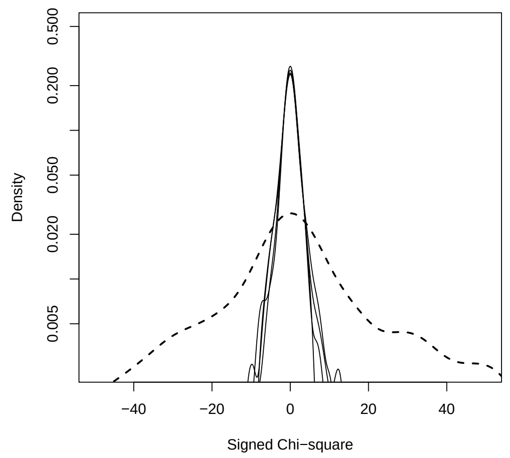


Figure 5: Distributions of random scores

Jensen and Cohen (2000) prove that, if the score of each member of a feature family is estimated with some variance, then the estimated score of the best member (the maximum score) will be a biased estimator of the feature's true score. In addition, that bias increases as the variance of the score distributions increases. Thus, the estimated score of features formed from objects with high linkage and autocorrelation (e.g., studios) will be more biased than those formed from objects with low linkage and autocorrelation (e.g., actors).

[2] The effective sample size estimated for studios is almost certainly too small, given that it is less than the total number of studios in the sample (128). The sampling distribution obtained for random attributes on studios had more density in the tails than the theoretical distribution for chi-square, and thus our similarity measures may not be adequate. We are exploring alternative methods for estimating effective sample size.

Results from the movie data clearly indicate high variance in estimated scores. Figure 7 shows score distributions based on multiple training/test splits of the movie data, where one set was used to select the best feature from each feature family, and the other set was used to obtain an unbiased score estimate. The scores vary widely, but features formed from studios have the highest variance. This experiment also indicates the competing pressures on feature selection algorithms in the face of high variance. Some random features on studios have variance sufficiently high to allow them to exceed the scores of weakly useful features on other objects. However, some non-random attributes on studios appear to form useful features, and any method for discounting high-variance features should not discard these.

4. Estimating Score Variance by Resampling

The first step toward correcting for high variance is to obtain accurate estimates of variance for each feature. In this section, we describe and test an approach to estimating score variance by bootstrap resampling.

4.1 Bootstrap Resampling

Bootstrap resampling is a technique for estimating characteristics of the sampling distribution of a given parameter by generating multiple samples by drawing, with replacement, from the original data as if it were the population (Noreen 1989). Each generated sample is called a *pseudosample* and contains as many instances as the original data set. Some instances in the original data set will occur multiple times in a given pseudosample, and others will not occur at all. Resampling can be used to estimate the variance of a parameter by estimating the parameter on hundreds of pseudosamples, and then finding the variance of the resulting distribution of scores.

To estimate the variance of a given score distribution using resampling, we draw links randomly and with replacement from all links in *P* until the number of links in the pseudosample is equal to the number in the original data. For example, to estimate variance for a relational attribute formed from studios, we would sample paths formed from *Made* links. Then we create objects based on the endpoints of the paths in the pseudosample. For example, we would create movie and studio objects based on the movies and studios that form the endpoints of the *Made* links in our pseudosample.

In most cases, we create a single object in response to many paths in the pseudosample with the same endpoint. For example, we would generally link many movies to a single studio object we have created for the pseudosample. In some cases, however, the degree of the resulting object in the pseudosample exceeds the degree of any similar object in the original data. In this case, we create an additional version of that object to keep linkage similar between the original sample and the pseudosamples. For example, movies in our data have single linkage to studios, thus we create extra movies in pseudosamples when the same link between movies and studios is sampled twice. The distribution of the scores calculated from many pseudosamples forms a distribution from which the variance can be estimated.

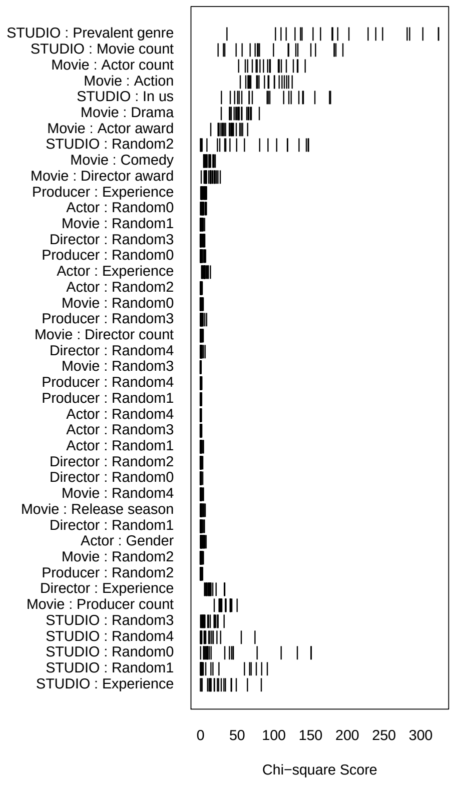


Figure 7: Scores of movie features and variance estimates

We evaluated the quality of pseudosamples by comparing their linkage, autocorrelation, and attribute distributions to the original sample. The measured quantities remain stable over all the pseudosamples and closely resemble the values in the original sample.

4.2 Using Resampled Estimates

While resampling can be used to estimate the variance of scores for particular features, the use of those estimates to improve feature selection remains an open problem. Figure 6 demonstrates the broad outlines of the problem. Given a set of scores for features and estimates of their sampling distributions, which features should be selected? In Figure 6, score *B* is clearly preferable to *A*, because it has both higher expected value and lower variance. However, scores *B* and *C* are not easily ranked because *C* has a higher expected value but also a higher variance.

We have tried two obvious ranking schemes without success. In the first, we ranked features based on their lower confidence limits (e.g., 5%). In the second, we grouped feature distributions into equivalence classes based on estimates of *prob(A>B)* for pairs of distributions. Features within equivalence classes were ranked based on variance. We evaluated these ranking schemes on randomly drawn subsets of movies and compared their rankings to the ranking on the full data set. We also conducted extensive simulations. Neither scheme significantly improved feature rankings. This issue of comparing distributions with unequal variance is a longstanding problem in statistics, and we are continuing to explore alternatives for improving feature selection.

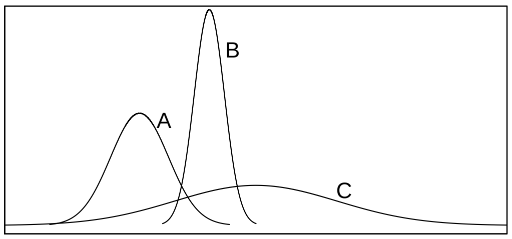


Figure 6: Example score distributions

Conclusions

Based on our work to date, substantial bias is likely to afflict relational learning algorithms that engage in feature selection in data sets with high linkage and autocorrelation. Some learning tasks avoid these conditions, either because the data consist of disconnected subgraphs (e.g., molecules) or because the data otherwise lack high linkage or autocorrelation. However, we have discovered high linkage and autocorrelation in data sets drawn from many domains, including web page analysis, fraud detection, and citation analysis. General-purpose algorithms for relational learning will need to address this source of bias.

We attribute the poor performance of our alternative ranking schemes to the fact that, given only the data, the individual score of a given feature remains the best estimate of its utility. Bootstrap resampling appears to provides accurate approximations to score distributions, but those distributions only indicate that the true score could be substantially lower or higher than the estimated score, with roughly equal probability. We suspect that future research to avoid feature selection bias will have to consider additional information, such as prior estimates of the true score. Given such information, priors on the true scores would receive different updates for the same set of data, because of the differing effective sample sizes for each feature. This may provide a better route to using the distribution information than our experiments to date.

Regardless of the shape of the eventual solution, the bias associated with linkage and autocorrelation indicates the importance of maintaining relational data representations, rather than propositionalizing data. Maintaining a relational data representation makes it possible to assess the statistical effects of linkage and autocorrelation, and to adjust for the resulting bias. In addition, as noted in section 2.2, maintaining relational representations allows inference procedures to exploit relational autocorrelation to improve the predictive accuracy of models.

Acknowledgments

Foster Provost and Hannah Blau provided valuable comments. Ross Fairgrieve prepared the movie data for analysis. The data used in this paper were drawn from the Internet Movie Database (www.imdb.com). This research is supported by The Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory (AFRL), Air Force Material Command, USAF, under agreements F30602-00-2-0597 and F30602-01-2-0566.

References

Cortes, C., D. Pregibon, and C. Volinsky (2001). Communities of Interest. *Proceedings of the Fourth International Symposium on Intelligent Data Analysis*.

Cressie, N. (1993). *Statistics for Spatial Data*. Wiley.

Dzeroski, S. and N. Lavrac, (Eds.) (2001). *Relational Data Mining*. Berlin: Springer.

De Raedt, L. (1998). Attribute-Value Learning versus Inductive Logic Programming: The Missing Links. *ILP '98*. 1-8. Springer

Flach, P. and N. Lachiche (1999). 1BC: a first-order Bayesian classifier. *ILP'99*. 92-103. Springer.

Getoor, L., N. Friedman, D. Koller, and A. Pfeffer (1999). Learning probabilistic relational models. *IJCAI'99*. 1300-1309.

Jensen, D. and P. Cohen (2000). Multiple comparisons in induction algorithms. *Machine Learning* 38:309-338.

John, G., R. Kohavi, and K. Pfleger (1994). Irrelevant features and the subset selection problem. *ICML'94*. 121-129.

Kleinberg, J. (1999). Authoritative sources in a hyperlinked environment. *Journal of the ACM* 46:604-632,

Muggleton, S. (2000). Learning Stochastic Logic Programs. *AAAI Workshop on Learning Statistical Models from Relational Data*, 36-41.

Noreen, E. (1989). *Computer Intensive Methods for Testing Hypotheses*. Wiley.

Neville, J. and D. Jensen (2000). Iterative Classification in Relational Data. *AAAI Workshop on Learning Statistical Models from Relational Data*, 42-49.

Sachs, L. (1982). *Applied Statistics.* Springer-Verlag.

Slattery, S., and Mitchell, T. (2000). Discovering test set regularities in relational domains. *ICML 2000. 895-902.*

Autocorrelation and Linkage Cause Bias in Evaluation of Relational Learners

David Jensen and Jennifer Neville

Department of Computer Science 140 Governors Drive University of Massachusetts, Amherst Amherst, MA 01003
{jensen|jneville}@cs.umass.edu

Two common characteristics of relational data sets — concentrated linkage and relational autocorrelation — can cause traditional methods of evaluation to greatly overestimate the accuracy of induced models on test sets. We identifythese characteristics, define quantitative measures of their severity, and explain how they produce this bias. We show how linkage and autocorrelation affect estimates of model accuracy by applying FOIL to synthetic data and to data drawn from the Internet Movie Database. We show how a modified sampling procedure can eliminate the bias.

Introduction

Accurate evaluation of learning algorithms is central to successful research in relational learning. The most common method for evaluating a learning algorithm is to partition a given data sample into training and test sets, construct a model using the training set, and evaluate the accuracy of that model on the test set. Separate training and test sets are used because of a widely observed bias when the accuracy of models is assessed on the original training set (Jensen & Cohen 2000).

In this paper, we show how dependence among the values of a class label in relational data can cause strong biases in the estimated accuracy of learned models when accuracy is estimated in this conventional way. In this section, we give a simple example of how estimated accuracy can be biased. In later sections, we define quantitative measures of concentrated linkage (L) and relational autocorrelation (C'),two common characteristics of relational data sets. We show how high values of L and C' cause statistical dependence between training and test sets, and we show how this dependence leads to bias in test set accuracy. In general, current techniques for evaluating relational learning algorithms do not account for this bias, although we discuss some special classes of relational data sets that are immune to this effect. We present a new family of sampling algorithms that can be applied to any relational data set, and show how it eliminates the bias.

These results indicate that accurate evaluation of relational learning algorithms will often require specialized evaluation procedures. The results also indicate that additional attention should be paid to identifying and using relational autocorrelation to improve the predictive accuracy of relational models. This paper is part of a larger study of the effects of linkage and autocorrelation on relational learning. A related paper (Jensen and Neville 2002) shows how linkage and autocorrelation affect feature selection in relational learning.

Statistical Analysis of Relational Data

Recent research in relational learning has focused on learning statistical models, including work on stochastic logic programming (Muggleton 2000), probabilistic relational models (Getoor et al. 1999), and relational Bayesian classifiers (Flach and Lachiche 1999). However, with the greater expressive power of relational representations come new statistical challenges. Much of the work on relational learning diverges sharply from traditional learning algorithms that assume data instances are statistically independent. The assumption of independence is among the most enduring and deeply buried assumptions of machine learning methods, but this assumption is contradicted by many relational data sets.

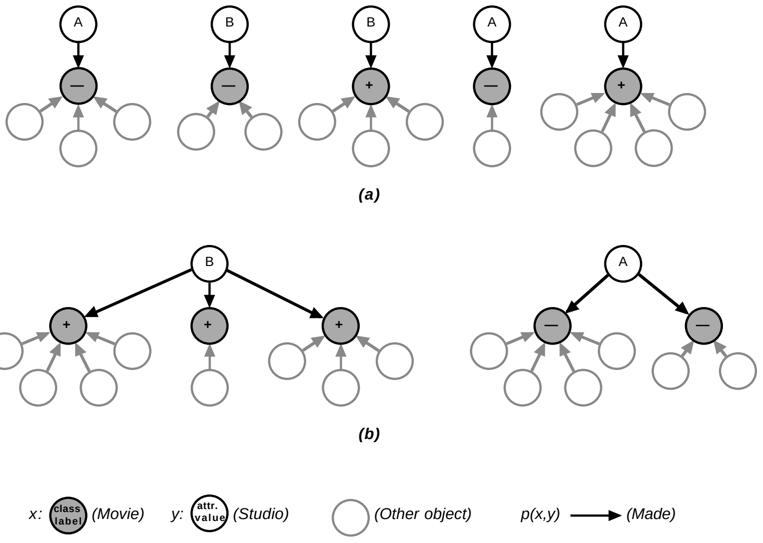


Fig. 1. Example relational data sets (a) five independent instances and (b) five dependent instances.

For example, consider the two simple relational data sets shown in Figure 1. In each set, instances for learning consist of subgraphs containing a unique object *x*, an object *y*, and one or more other objects. Objects *x* contain a class label and objects *y* contain an attribute that will be used to predict the class label of *x*. Figure 1a shows a data set where objects *x* and *y* have a one-to-one relationship and where the class labels on instances are independent. Figure 1b shows instances where objects *x* and *y* have a many-to-one relationship and where the class labels are dependent.

We spend the majority of this paper considering data sets similar in structure to Figure 1b, where each subgraph consists of multiple relations and each relation may produce statistical dependencies among the instances. For simplicity, all relations in Figure 1 are binary, but the statistical effects we investigate affect a wider range of tasks and data representations.

Simple Random Partitioning

Perhaps the most widely used evaluation technique in machine learning and data mining is the partioning of a data sample into training and test sets. Most sampling in machine learning and data mining assumes that instances are independent. In contrast, this paper examines situations where instances are not independent. Methods for sampling relational data are not well understood. In the relatively few cases where researchers have considered special methods for sampling relational data for machine learning and data mining, they have often relied on special characteristics of the data. For example, some researchers have exploited the presence of naturally occurring, disconnected subsets in the population, such as multiple websites without connections among the sites (e.g., Craven et al. 1998). We wish to evaluate classification models that operate over completely connected graphs. There is also a small body of literature on sampling in relational databases (e.g., Lipton et al. 1993), but this work is intended to aid query optimization while our interest is to facilitate evaluation of predictive models.

The most common sampling technique — *simple random partitioning* — has been taken from propositional settings and adapted for use in relational sets such as those in Figure 1. We define simple random partitioning with respect to two sets of objects X and Y and a set of paths P such that the relation $p(x,y)$ holds. We presume that objects X contain class labels and that objects X and Y both contain attributes relevant to classifying objects X. Paths are composed of k links and $k-1$ intervening objects, where $k \geq 1$. Each path represents a series of relations linking an object in X to an object in Y. For example consider the path linking two movies, m_1 and m_2, made by the same studio. The path is formed from two *Made* links, $Made(m_1,s_1)$ and $Made(m_2,s_1)$. For convenience we treat all links as undirected in order to refer to meaningful sequences of relationships as paths. We assume that paths in P are unique with respect to a given (x,y) pair; if two or more paths between x and y exist in the data, they are collapsed to a single element of P.

> **Definition:** *Simple random partitioning* divides a sample S of relational data into two subsamples S_A and S_B. The subsamples are constructed by drawing objects X from S without replacement and without reference to paths P and objects Y in S. When an individual object $x \in X$ is placed into a subsample, some set of objects $\{y_1, y_2, \ldots y_n\}$, such that $p(x,y_i)$, are also placed into the subsample if those objects are not already present. The object sets X_A and X_B are mutually exclusive and collectively exhaustive of objects X in S, but other objects Y may appear in both S_A and S_B.

Such a techique for creating training and test sets seems a logical extension of techniques for propositional data. However, it leaves open the possibility that a subset of objects *Y* may fall into both the training and test set, creating some type of dependence between the training and test sets.

An Example of Test Set Bias

Given that instances in relational data may share some objects, and thus not be independent, we should examine how such relational structure could affect accuracy estimates made using training and test sets. Below we show how relational structure and dependence among values of the class label can bias estimates of the accuracy of induced models.

We created data sets about movies and analyzed them using FOIL (Quinlan 1990). Specifically, we created and analyzed simple relational data sets whose relational structure was drawn from the Internet Movie Database (www.imdb.com). We gathered a sample of all movies in the database released in the United States between 1996 and 2001 for which we could obtain information on box office receipts. In addition to 1382 movies, the data set also contains objects representing actors, directors, producers, and studios.[1] In all, the data set contains more than 40,000 objects and almost 70,000 links. The data schema is shown in Figure 2.

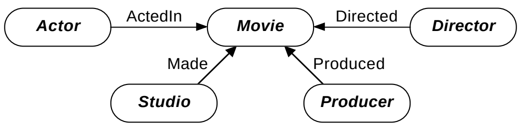


Fig. 2. A general data schema for the movie data sets.

For most of the experiments reported in this paper, we limited analysis to just two classes of objects — movies and studios. This allowed us to greatly reduce the overall size of training and test sets, thus making them feasible to analyze using FOIL. This also focused the experiments on precisely the phenomena we wished to study, as discussed below. Details about the data representation are given in the appendix.

We created a learning task using an attribute on movies — opening-weekend box office receipts. We discretized the attribute so that a positive value indicates a movie with more than \$2 million in opening weekend receipts (*prob(+)=0.55*). We call this discretized attribute *receipts* and use it as a binary class label. We also created ten random attributes on studios. The values of these attributes were randomly drawn from a uniform distribution of five values, and thus were independent of the class label. Figure 3 shows the schema with movies, studios, and their attributes.

We used simple random partitioning to create (approximately) equal-sized training and test sets. This divides our sample of 1382 movies in two subsamples, each containing approximately 690 movies and their affiliated studios. Each movie appears in only one sample. Each affiliated studio might appear in one or both subsamples,

[1] Each of the movies is related to one primary studio. For movies with more than one associated studio, we chose the U.S. studio with highest degree to be the primary, for movies without any U.S. studios we chose the studio of highest degree to be the primary.

and would never appear more than once in any one subsample. However, a single studio object can be linked to many movies in a given subsample.

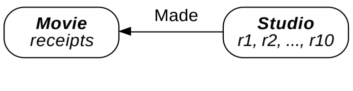


Fig. 3. A simplified data schema, with attributes, for the artificial data sets used for experiments in this section. Attributes denoted *r1* through *r10* are random attributes.

Given these data sets, we evaluated the ability of FOIL to learn useful models in the traditional way. We ran FOIL on the training set and evaluated the accuracy of the resulting models on the test set. Given that attributes on studios were created randomly, the expected error for the models constructed exclusively from those attributes should equal the default error (0.55). Deviations from this error represent a bias, which can be measured by subtracting the measured error $\hat{e}$ from the theoretical error e. Positive bias over many trials indicates that the test set accuracy is systematically lower than the theoretical error.

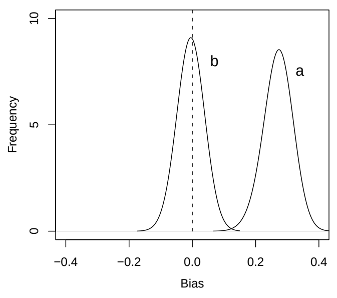


Fig. 4. Distribution of test set bias for FOIL models constructed from random attributes using (a) the actual class labels and (b) randomized class labels. The distributions were smoothed using a bandwidth parameter of 0.4.

Figure 4 shows two distributions of bias estimated from 50 different training and test set partitions. The rightmost distribution (a) results from the experiment described above. The bias is substantially larger than zero, indicating that the measured error of FOIL rules on the test set is much lower than the default. The leftmost distribution (b) results from running the same experiment, except that the values of the class label on movies (*receipts)* are randomly reassigned before each trial. This distribution has almost precisely the expected bias of zero.

These experimental results raise obvious questions: Why does the algorithm appear to learn from random features formed from studios when the actual class label is used, but not when the values of that class label are randomly assigned? What does this result tell us about evaluating relational learners in general?

Linkage, Autocorrelation, and Overfitting

Our analysis indicates that biases like that shown in Figure 4 result from the confluence of three common phenomena in relational learning — *concentrated linkage*, *relational autocorrelation*, and *overfitting*. Concentrated linkage and relational autocorrelation create statistical dependence among instances in different samples, and overfitting exploits that dependence in ways that lead to bias in test set accuracy. We define concentrated linkage and relational autocorrelation formally below. Informally, concentrated linkage occurs when many objects are linked to a common neighbor, and relational autocorrelation occurs when the values of a given attribute are highly uniform among objects that share a common neighbor. Overfitting is familiar to machine learning researchers as the construction of complex models that identify unique characteristics of the training set rather than statistical generalizations present in the population of all data.

Much of the text of this section is drawn from an earlier paper (Jensen & Neville 2002). However, the definitions are so central to understanding the experiments in later sections that we present this material again rather than attempting to summarize.

Concentrated Linkage

We define concentrated linkage *L(X,P,Y)* with respect to the same conditions as simple random partioning — two sets of objects *X* and *Y* and a set of paths *P* such that *p(x,y)*.

Definition: D_{yX} is the degree of an object *y* with respect to a set of objects *X*. That is, the number of $x \in X$ such that $p(x,y) \in P$. For example, D_{yX} might measure, for a given studio *y*, the number of movies (*X*) it has made. ■

Definition: *Single linkage* of *X* with respect to *Y* occurs in a data set whenever, for all $x \in X$ and $y \in Y$:

$$D_{xY} = 1 \quad \textit{and} \quad D_{yX} \geq 1 \qquad ■$$

In these cases, many objects in *X* (e.g., movies) connect to a single object in *Y* (e.g., a studio). We use single linkage as an important special case in future discussions.

Definition: The *concentrated linkage L(x,X,P,Y) of an individual object x* (e.g., a movie) that is linked to objects *Y* (studios) via paths *P* is:

$$L(x,X,P,Y) = \sum_{\substack{y\ s.t.\\ p(x,y)\in P}} \frac{(D_{yX}-1)}{D_{yX}} \Bigg/ {D_{xY}}^2 \qquad ■$$

the quantity $(D_{yX} - 1)/D_{yX}$ within the summation is zero when the D_{yX} is one, and asymptotically approaches one as degree grows, and thus is a reasonable indicator of *L(x,X,P,Y)*, given single linkage of *x* with respect to *Y*. Because *x* may be linked to multiple nodes in *Y*, we define the average across all nodes y_i linked to *x*, and divide by an additional factor of D_{xY} to rate single linkage more highly than multiple linkage.

Definition: The *concentrated linkage L(X,P,Y) of a set of objects X* (e.g., all movies) that are linked to objects *Y* is:

$$L(X,P,Y) = \sum_{x \in X} \frac{L(x,X,P,Y)}{|X|} \quad \blacksquare$$

Given particular types of linkage, *L* can be calculated analytically from the sufficient statistics |*X*| and |*Y*|. For example, in the case of single linkage of *X* with respect to *Y*, *L = (*|*X*|-|*Y*|*)/*|*X*|. For example, the data set shown in Figure 1b exhibits single linkage, so *L(X,P,Y)* = 0.60. Propositional data also display single linkage, and because |*X*|=|*Y*|, *L(X,P,Y) = 0.* Calculations of several types of linkage are shown for the movie data in Table 2.

Table 2: Linkage in the movie data

Linkage Type	Value
L(Movie, Made, Studio)	0.91
L(Movie, Directed, Director)	0.23
L(Movie, Produced, Producer)	0.08
L(Movie, ActedIn, Actor)	0.01

In addition to the movie data, we have encountered many other instances of concentrated linkage. For example, while studying relationships among publicly traded companies in the banking and chemical industries, we found that nearly every company in both industries uses one of only seven different accounting firms. In work on fraud in mobile phone networks, we found that 800 numbers, 900 numbers, and some public numbers (e.g., 911) produced concentrated linkage among phones. Concentrated linkage is also common in other widely accessible relational data sets. For example, many articles in the scientific literature are published in single journals and many basic research articles are cited in single review articles. On the Web, many content pages are linked to single directory pages on sites such as Yahoo and Google.

Correlation and Autocorrelation

We will define relational correlation *C(X,f,P,Y,g)* with respect to two sets of objects *X* and *Y*, two attributes *f* and *g* on objects in *X* and *Y*, respectively, and a set of paths *P* that connect objects in *X* and *Y*.

Definition: *Relational correlation C(X,f,P,Y,g)* is the correlation between all pairs *(f(x),g(y))* where *x*∈*X*, *y*∈*Y* and *p(x,y)*∈*P*. ■

Given the pairs of values that these elements define, traditional measures such as information gain, chi-square, and Pearson's contingency coefficient can be used to assess the correlation between values of the attributes *f* and *g* on objects connected by paths in *P*. The range of *C* depends on the measure of correlation used.

We can use the definition of relational correlation *C(X,f,P,Y,g)* to define relational *autocorrelation* as the correlation between the same attribute on distinct objects belonging to the same set.

Definition: *Relational autocorrelation* C' is:

$C'(X, f, \mathrm{P}) = \mathrm{C}(X, f, \mathrm{P}, X, f)$ where For All $p(x_i, x_j) \in \mathrm{P}$ $x_i \neq x_j$

For example, C' could be defined with respect to movie objects, the attribute *receipts* on movies, and paths formed by traversing *Made* links that connect the movies to an intervening studio.

If the underlying measure of correlation varies between zero and one, then C'=1 indicates that the value of the attribute for a specific node x_i is always equal to all other nodes x_j reachable by a path in P. When C'=0, values of $f(X)$ are independent. Table 3 gives estimates of relational autocorrelation for movie receipts, linked through studios, directors, producers, and actors. For a measure of correlation, Table 3 uses Pearson's corrected contingency coefficient (Sachs 1992), a measure that produces an easily interpreted value between zero and one. Autocorrelation is fairly strong for all object types except actors.

In addition to the movie data, we have encountered many other examples of high relational autocorrelation. For example, in our study of publicly traded companies, we found that when persons served as officers or directors of multiple companies, the companies were often in the same industry. Similarly, in biological data on protein interactions we analyzed for the 2001 ACM SIGKDD Cup Competition, the proteins located in the same place in a cell (e.g., mitochondria or cell wall) had highly autocorrelated functions (e.g., transcription or cell growth). Such autocorrelation has been identified in other domains as well. For example, fraud in mobile phone networks has been found to be highly autocorrelated (Cortes et al. 2001). The topics of authoritative web pages are highly autocorrelated when linked through directory pages that serve as "hubs" (Kleinberg 1999). Similarly, the topics of articles in the scientific literature are often highly autocorrelated when linked through review articles.

Table 3: Autocorrelation in the movie data

Autocorrelation Type	Value
C'(*Movie,Receipts,Made | Studio | Made*)	0.47
C'(*Movie,Receipts,Directed | Director | Directed*)	0.65
C'(*Movie,Receipts,Produced | Producer | Produced*)	0.41
C'(*Movie,Receipts,ActedIn | Actor | ActedIn*)	0.17

Note: The notation $a|x|b$ to denote paths with links of type a and b and intervening objects of type x.

We define relational autocorrelation in a similar way to existing definitions of temporal and spatial autocorrelation (see, for example, Cressie 1993). Autocorrelation in these specialized types of relational data has long been recognized as a source of increased variance. However, the more general types of relational data commonly analyzed by relational learning algorithms pose even more severe challenges because the amount of linkage can be far higher than in temporal or spatial data and because that linkage can vary dramatically among objects.

Relational autocorrelation represents an extremely important type of knowledge about relational data, one that is just beginning to be explored and exploited for learning statistical models of relational data (Neville and Jensen 2000; Slattery and

Mitchell 2000). Deterministic models representing the extreme form of relational autocorrelation have been learned for years by ILP systems. By representing and using relational autocorrelation, statistical models can make use of both partially labeled data sets and high-confidence inferences about the class labels of some nodes to increase the confidence with which inferences can be made about nearby nodes.

However, as we show below, relational autocorrelation can also greatly complicate learning of all types of relational models. As we seek to represent and use relational autocorrelation in statistical models of relational data, we will need to adjust for its effects when evaluating more traditional types of features in these models.

Effects of Linkage, Autocorrelation, and Overfitting on Bias

The results reported so far for concentrated linkage and relational autocorrelation provide important clues to the behavior shown in Figure 4. Studios are the objects in the movie data that have the highest combination of concentrated linkage and relational autocorrelation. In this section, we show that, if linkage and autocorrelation are both high for a single type of object, and an algorithm produces overfitted models that use attributes on those objects, then the test set error will be biased.

Dependent Training and Test Sets

Given the definition of simple random partitioning in the introduction, we can examine the statistical dependence of subsamples it produces. We define $prob_{ind}(A,B)$ to be the probability that subsamples A and B are independent.[2] Further, we define $prob_{ind}(A,B|y)$ to be the probability that subsamples A and B are independent with respect to a specific object $y \in Y$, that is where A and B are independent with respect to objects $x_i \in X$ such that $p(x_i,y) \in P$.

Theorem: Given simple random partitioning of a relational data set S with single linkage and $C'=1$:

$$prob_{ind}(A,B) \rightarrow 0 \quad as \quad L \rightarrow 1.$$

Proof: First, consider a sample S composed of independent subgraphs such as those shown in Figure 1a. In such a sample, $L=0$ because no x is linked to more than one y, and $prob_{ind}(A,B)=1$ because no object x can fall into more than one sample and there is no dependence among objects x_1 and x_2 because $L = 0$.

Now consider the effect of increasing D_{yX}, the degree of an object y (e.g., a studio) with respect to objects X (e.g., movies). For notational convenience, $d=D_{yX}$. Increasing d necessarily increases L for a single object x, because for single linkage $L(x,X,P,Y)=(d-1)/d$. For any one object y, the samples A and B are not independent if both contain an object x_i, such that $p(x_i,y)$. Thus:

[2] In this paper, we consider the effects of dependence between instances in different subsamples, but not the effects of dependence among objects within the *same* subsample. The latter topic is covered in another recent paper (Jensen and Neville 2002).

$$prob_{ind}(A,B \mid y) = b(0,d,p) + b(d,d,p) \qquad \textbf{(1)}$$

where $b(m,n,p)$ denotes the value of the binomial distribution for m successes in n trials, where each trial succeeds with probability p. Here, for example, $b(d,d,p)$ is the probability that a given sample (e.g., A) will contain all of the d movies linked to a given studio y. In the case of simple random partitioning into equal-sized samples, $p=0.5$. For high values of d (many objects x are connected to a single object y), $prob_{ind}(A,B|y)$ approaches zero. For $d=2$, $prob_{ind}(A,B|y)=0.5$. That is, there is only a 50% probability that both objects x connected to y will end up in the same sample. For $d=3$, $prob_{ind}(A,B|y)=0.25$; for $d=10$, $prob_{ind}(A,B|y)=0.002$.

Given that S contains many objects Y, the probability of independence for *all* objects y becomes vanishingly small. Specifically:

$$prob_{ind}(A,B) = \prod_{y} prob_{ind}(A,B \mid y) \qquad \textbf{(2)}$$

For even small samples, $prob_{ind}(A,B)$ goes quickly to zero as L increases. For example, given a sample of 50 instances of X, if $d=2$ $(L=0.5)$, then $prob_{ind}(A,B)=3.0x10^{-8}$. For $d=5$ $(L=0.8)$, $prob_{ind}(A,B)=9.1x10^{-13}$. For large samples and L>0, $prob_{ind}(A,B) \approx 0$. ■

Not only is the probability of *any* dependence between A and B high, but the degree of dependence is very likely to be high. For example, the binomial distribution can be used to derive the expected number of objects x_1 in sample A with a matching object x_2 in B such that $p(x_1,y) \in P$ and $p(x_2,y) \in P$ for some $y \in Y$ with degree $d=D_{yX}$. The maximum and expected number of matched pairs of dependent instances is:

$$Max(pairs) = \lfloor d/2 \rfloor \qquad \textbf{(3)}$$

$$E(pairs) = \sum_{i=\square}^{d} \min(i, d-i)\, b(i,d,0.5)$$

For example, for $d=5$, the maximum number of pairs is $Max(pairs)=2$ and the expected number is $E(pairs)=1.56$. For $d=10$, $Max(pairs)=5$ and $E(pairs)=3.77$.

How Bias Varies with Autocorrelation, Linkage, and Overfitting

Given dependent training and test sets, it is relatively easy to see how overfitted models can cause bias in test set accuracy. One way of characterizing the observed behavior is that it represents a relational version of the "small disjuncts" problem (Holte, Acker, & Porter 1989). This problem arises in propositional learning when overfitted models parse the instance space into sets ("disjuncts") containing only a few data instances. For example, when a decision tree is pathologically large, its leaf nodes can apply to only a single instance in the training set. Such models perform as lookup tables that map single instances to class labels. They achieve very high accuracy on training sets, but they generalize poorly to test sets, when those test sets are statistically independent of the training set.

In the relational version of the small disjuncts problem, models use relational attributes to parse the space of objects *y* into sets so small that they can uniquely identify one such object (e.g., a single studio). If that object is linked to many objects *x* where a single class predominates (e.g., *receipts* = +), then a model that uniquely identifies that object *y* can perform well on the training data. If that *y* also appears in the test data, then the model can perform well on test data.

Indeed, such overfitting is made more likely by high autocorrelation among the values of the class label *within* a given training set. If several objects *X* in a training set are all linked to a single object *y*, and if the class labels of the objects *X* are highly correlated, then it is more likely that a learning algorithm will create a model with components intended to predict precisely these instances than if only a single instance had this combination of attribute values and class label.

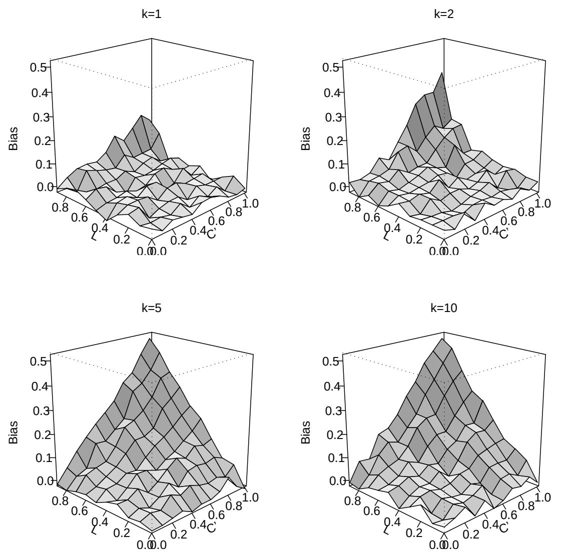


Fig. 5. Bias increases with linkage (*L*), autocorrelation (*C'*), and number of random attributes (*k*). Each point represents the average of 20 trials.

As noted above, relational autocorrelation represents an extremely important type of knowledge about relational data, one that can be exploited to improve accuracy (Neville and Jensen 2000; Slattery and Mitchell 2000). However, it can also fool algorithms and evaluation techniques not designed to account for its effects.

For example, consider the results shown in Figure 5. The graphs show how the bias varies across a wide range of linkage (*L*), autocorrelation (*C'*), and potential for overfitting. To alter this latter characteristic of learning algorithms, we varied the number of attributes *k* from one to ten. In each trial, we created synthetic data sets with 200 objects *X* and specified values of autocorrelation and single linkage with objects *Y*. Each object *x* was given a class label drawn from a binary uniform distribution. Each object *y* was given *k* attributes, each with a value drawn from a five-valued uniform distribution. This sample was then divided into equal-sized training and test sets using simple random partitioning. We applied FOIL to the training set, and then evaluated the error of the resulting rules on the test set. For each combination of *L*, *C'*, and *k*, we ran 20 trials and averaged the bias.

For a given number of attributes *k*, bias increases dramatically with increasing linkage *L* and autocorrelation *C'*. For high values of *k*, *L*, and *C'*, bias is maximal (0.5). However, even moderate values of *k*, *L*, and *C'* produce substantial bias, confirming the results in the first section.

It is important to note that in the experiments presented above, these overfitted models are *not* learning any general knowledge about autocorrelation and linkage. For example, the FOIL rules learned for the experimental results depicted in Figure 4 contain only clauses of the form:

```
    receipts(A) :- made-by(A,B), studio-attributes(B,C,D,E,F).
```

That is, they exclusively relate attributes of studios to *receipts* of movies linked to them. As noted previously, linkage and autocorrelation represent an important type of knowledge that could be exploited by a relational learning algorithm. However, most relational learning algorithms either cannot or do not learn probabilistic models of this form. The rules formed by FOIL on the synthetic data used in Figure 5 are of a similar form. The results in Figure 5 show that large bias that can result when algorithms learn overfitted models from data with strong linkage and autocorrelation.

Subgraph Sampling

Fortunately, this bias can be eliminated by a relatively small change to the procedure for creating training and test sets. *Subgraph sampling* guarantees that an object *y* and corresponding objects *X* appear only within a single subsample. This confines any autocorrelation among the class labels of objects *X* to a single subsample, and thus removes the dependence between subsamples due to concentrated linkage and relational autocorrelation.

We first introduced subgraph sampling of relational data in an earlier paper (Jensen and Neville 2001). However, we lacked a full understanding of the causes of dependence between subsamples, and we proposed an extreme form of subgraph sampling that eliminated all possible duplication of objects between subsamples, even when class labels were not autocorrelated through all types of objects. Here we propose a form of subgraph sampling that is far more conservative.

First, consider the special case where linkage and autocorrelation are high for only one type of object *y*, and that object exhibits single linkage with objects *X*. For example, in the movie data, only studios exhibit both high linkage and high autocorrelation; other types of objects (actors, directors, and producers) have fairly low values for one or both quantities. In addition, studios exhibit single linkage with movies. In this special case, we can partition a sample *S* based on the objects *Y* (e.g., studios), and then place all objects *X* in the same subsample as their corresponding *y*.

A more general partitioning algorithm first assigns objects *X* to prospective samples, and then incrementally converts prospective assignments to permanent assignments only if the corresponding objects *Y* for the given *x* are disjoint from other objects *Y* already assigned to subsamples other than the prospective subsample of *x*. In contrast to the approach we proposed earlier (Jensen and Neville 2001), the objects *Y* considered during sampling should only be those through which linkage and autocorrelation is high.[3]

One feature of the general algorithm is worthy of special note — the random assignment of objects *X* to "prospective" subsamples. The algorithm either makes a prospective assignment permanent, or discards the object. An alternative algorithm would search for an assignment of objects to permanent subsamples that maximizes the number of objects assigned to each subsample, thus maximizing the size of subsamples. However, this approach can induce another form of statistical dependence among subsamples. Consider how such an "optimizing" algorithm would behave when confronted with a data set consisting of two disjoint (or nearly disjoint) sets of relational data. One subsample would be filled entirely with objects from one disjoint set, and another would be filled with objects from the other set. If the statistical characteristics of one of the disjoint sets did not mirror the characteristics of the other, then accuracy estimates of learned models would be biased downward.

Subgraph sampling resembles techniques that construct samples from a small number of completely disconnected graphs. For example, some experiments with WEBKB (Slattery and Mitchell 2000) train classification models on pages completely contained within a single website, and then test those models on pages from another website with no links to the training set. This approach exploits a feature of some websites — heavy internal linkage but few external links. Similarly, some work in ILP constructs samples from sets of completely disconnected graphs (e.g., individual molecules or English sentences) (Muggleton 1992). This approach are possible only when the domain provides extremely strong natural divisions in the graphs, and this approach is only advisable where the same underlying process generated each graph. In contrast, subgraph sampling can be applied to data without natural divisions. Where they exist, subgraph sampling will exploit some types of natural divisions. Where they do not exist, logical divisions can be created that preserve the statistical independence among samples.

Subgraph Sampling Eliminates Bias

In this section, we show how subgraph sampling eliminates the bias caused by linkage, autocorrelation, and overfitting. First, we replicate the experiments that produced Figure 4. However, rather than learning models for a randomized class label, we learn models on the original class label, but with samples produced by subgraph sampling. The results are shown in Figure 6. As before, the bias associated with simple random partitioning is high. However, the distribution of bias for subgraph sampling (b) has a mean bias near zero.

[3] What is considered "high" would vary somewhat by the desired precision of the estimated accuracy. This is a topic for future work.

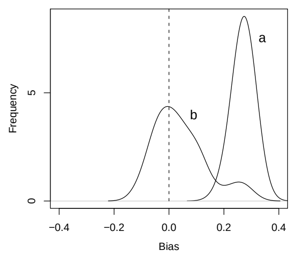


Fig. 6. Bias of FOIL models using non-random attributes for (a) simple random partitioning and (b) subgraph sampling.

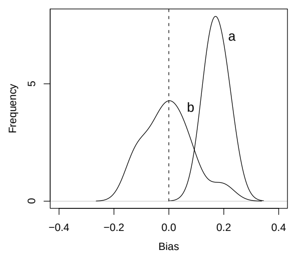


Fig. 7. Bias in FOIL models using non-random attributes for (a) simple random partitioning and (b) subgraph sampling.

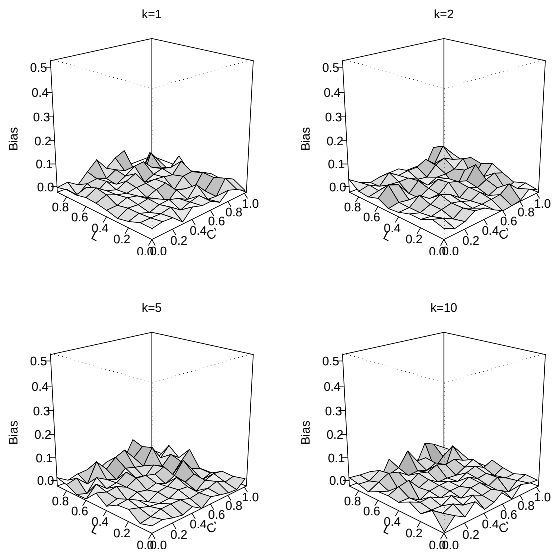


Fig. 8. Subgraph sampling with maximal separation eliminates bias at all levels of linkage (*L*), autocorrelation (*C'*), and number of random attributes (*k*).

The results in figures 4 and 6 were obtained using completely random attributes artificially generated on studios. However, similar results are obtained if we learn from attributes generated from the real characteristics of studios. The results in Figure 7 were generated by learning models with four attributes on studios. The attributes are the first letter of the studio name, the decade in which the studio was founded, the number of letters in the studio name (discretized to 10 unique values), and a binary attribute indicating whether the studio is located in the U.S. As before, the bias is high for simple random partitioning. Bias for both distributions used the mean of the distribution for subgraph sampling as an estimate of true error. These results confirm that the bias does not result from some peculiarity in the generation of random attributes, but rather results from dependence between the training and test sets

These results only indicate bias for a single combination of *L*, *C'*, and degree of overfitting. Figure 8 shows the results of more systematic variation of these quantities. The figure was produced from the same experiments as Figure 5, except the training and test sets were constructed by subgraph sampling rather than simple random partitioning. The result is extremely low bias across the full range of values of *L*, *C'*, and *k*..

Conclusions and Future Work

Concentrated linkage and relational autocorrelation can cause strong bias in the test set accuracy of induced models. In this paper, we demonstrate the bias using FOIL, so that other researchers can easily replicate and extend our experiments, but we have also observed this phenomenon in our own algorithms for relational learning. Fortunately, the bias associated with linkage and autocorrelation can be corrected by using subgraph sampling in preference to simple random partitioning.

While some special classes of relational data naturally allow subgraph sampling, relational learning methods will increasingly encounter data in which this bias arises, as we extend our work to more general classes of relational data, including networks of web pages, bibliographic citations, financial transactions, messages, biochemical interactions, computers, supervisory relationships, and social interactions.

This work also emphasizes the need to pursue research on relational learning techniques that exploit relational autocorrelation to enhance the predictive power of relational models. Additional work should also investigate methods to estimate the bias associated with specific levels of autocorrelation and linkage, and to search for classes of objects that exhibit those degrees of linkage and autocorrelation, so more automated approaches to subgraph sampling can be devised.

Acknowledgments

Foster Provost and Hannah Blau provided valuable comments on an earlier version of this work and Ross Fairgrieve prepared the movie data for analysis. The data used in this paper were drawn from the Internet Movie Database (www.imdb.com) and several experiments used FOIL, by Ross Quinlan. This research is supported by The Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory (AFRL), Air Force Material Command, USAF, under agreements F30602-00-2-0597 and F30602-01-2-0566.

References

Cortes, C., D. Pregibon, and C. Volinsky (2001). Communities of Interest. *Proceedings Intelligent Data Analysis 2001.*

Craven, M., DiPasquo, D., Freitag, D., McCallum, A., Mitchell, T., Nigam, K., & Slattery, S. (1998). Learning to extract symbolic knowledge from the World Wide Web. *Proceedings of the Fifteenth National Conference on Artificial Intellligence (*pp. 509-516). Menlo Park: AAAI Press.

Cressie, N. (1993). *Statistics for Spatial Data*. Wiley.

Flach, P. and N. Lachiche (1999). 1BC: a first-order Bayesian classifier. *ILP'99*. 92-103. Springer.

Getoor, L., N. Friedman, D. Koller, and A. Pfeffer (1999). Learning probabilistic relational models. In *IJCAI'99*. 1300-1309.

Holte, R., Acker, L., & Porter, B. (1989). Concept learning and the accuracy of small disjuncts. *Proceedings of the 11th International Joint Conference on Artificial Intelligence* (pp. 813-818). Detroit: Morgan Kaufmann.

Jensen, D. and P. Cohen (2000). Multiple comparisons in induction algorithms. *Machine Learning* 38:309-338.

Jensen, D. and J. Neville (2001). Correlation and sampling in relational data mining. *Proceedings of the 33rd Symposium on the Interface of Computing Science and Statistics.*

Jensen, D. and J. Neville (2002). Linkage and autocorrelation cause bias in relational feature selection. *Machine Learning: Proceedings of the Nineteenth International Conference*. Morgan Kaufmann.

John, G., R. Kohavi, and K. Pfleger (1994). Irrelevant features and the subset selection problem. *ICML'94*. 121-129.

Kleinberg, J. (1999). Authoritative sources in a hyper-linked environment. *Journal of the ACM* 46:604-632,

Lipton, R., Naughton, J., Schneider, D., & Seshadri, S. (1993). Efficient sampling strategies for relational database operations. *Theoretical Computer Science*, 116, 195-226.

Muggleton, S. (Ed) (1992). *Inductive Logic Programming*. Academic Press

Muggleton, S. (2000). Learning Stochastic Logic Programs. AAAI Workshop on Learning Statistical Models from Relational Data, 36-41.

Noreen, E. (1989). *Computer Intensive Methods for Testing Hypotheses*. Wiley.

Neville, J. and D. Jensen (2000). Iterative Classification in Relational Data. AAAI Workshop on Learning Statistical Models from Relational Data, 42-49.

Quinlan, J. R. (1990), Learning logical definitions from relations. *Machine Learning* 5:239-266.

Sachs, L. (1982). *Applied Statistics*. Springer-Verlag.

Slattery, S. & Mitchell, T. (2000). Discovering test set regularities in relational domains. *Proceedings of the 17th International Conference on Machine Learning*. San Francisco: Morgan Kaufmann.

Appendix

The movie data were coded for input to FOIL as follows: Each studio attribute was specified as an unordered discrete type with all attribute values flagged as a theory constants. Unordered type specifications also define movies and studios, with 1382 unique labels for the movies and 128 unique labels for the studios respectively.

The input contains one target relation and two background relations:

```
receipts(movie)
made-by(studio)
studio-attributes(studio, first-char, name-len, in-us, decade)
```

The target relation described above for movie receipts contains both positive and negative examples. The two background relations contain only positive examples; one specifying the relationships between movies and studios, and the other specifying attribute values associated with each studio.

Learned clauses were similar in form to:

```
receipts(A) :- made-by(A,B), studio-attributes(B,C,D,E,F).
```

All experiments used the current version of FOIL (foil6.sh) obtained from: <http://www.cse.unsw.edu.au/~quinlan/>. Arguments to FOIL specified that negative literals were not to be considered and the minimum accuracy of any clause considered was at least 70%. Other than these two modifications, FOIL's default settings were used.

J. Neville and D. Jensen (2002). Supporting relational knowledge discovery: Lessons in architecture and algorithm design.
Papers of the ICML 2002 Workshop on Data Mining Lessons Learned.

Supporting Relational Knowledge Discovery: Lessons in Architecture and Algorithm Design

Jennifer Neville JNEVILLE@CS.UMASS.EDU
David Jensen JENSEN@CS.UMASS.EDU
Knowledge Discovery Laboratory, Department of Computer Science, University of Massachusetts, 140 Governors Drive, Amherst, MA 01003 USA

Abstract

This paper discusses a few of the lessons we have learned developing a relational knowledge discovery system. The relationships among data instances in relational data provide extra information for "mining." This additional information has the potential to greatly improve the quality of learned models. However, the dependencies among instances in the data also introduce new statistical challenges for learning algorithms. Relational data provide an ideal environment in which to examine a central challenge of knowledge discovery – its "chicken and egg" character. Data representation can impair the ability to learn important knowledge, but knowing the "right" data representation often requires just that knowledge. With relational data, representation is often a choice; many alternate views of the data provide abundant fodder for reasoning about transformations. In light of this, we discuss representation and design choices that support a co-evolutionary process of knowledge discovery and data transformation in relation data.

1. Introduction

Relational knowledge discovery is a new frontier of data mining that is just starting to be explored. Relational data offer a wealth of previously untapped information to exploit during the discovery process. However, along with new opportunities for discovery, the distinctive characteristics of relational data also present several unique challenges. Relationships in the data represent interdependencies among instances and these dependencies can make it difficult to learn accurate probabilistic models of the data. Also, there are often many alternative ways to represent any given set of relational data. Knowing the "right" representation can be crucial to the discovery process but often this knowledge is not known a priori and needs to be learned during analysis. In order to exploit the opportunities offered by relational data, these issues should inform the design of both the architecture and the algorithms of relational knowledge discovery systems.

The lessons we report have been amassed throughout the process of developing and applying PROXIMITY, a system for relational knowledge discovery[1]. PROXIMITY is a set of tools that operate on a graph database designed to support the process of knowledge discovery in relational data. The tools of PROXIMITY provide methods for an iterative process of attribute creation, model learning and inference. One of the goals of the system is to support a transformative approach to knowledge discovery, which will be discussed in a later section.

The majority of data routinely captured by businesses and organizations are relational, yet over the past decade most data mining research has focused on propositional data. Relational data offer unique opportunities to boost the accuracy of learned models and improve the quality of decision-making if the algorithms can learn effectively from the additional information the relationships provide. Recent efforts to modify traditional data mining techniques for relational data include modified Bayes classifiers, decision trees and association rules (Dzeroski & Lavrac 2001). These algorithms have been successfully applied across a range of applications but the explosive growth of structured data suggests more work is needed in this direction.

Over the last two decades much of the work in the machine learning and knowledge discovery communities has focused on non-relational data, where instances are assumed to be identical and independently distributed (i.i.d.). Relational data violate this assumption. Relationships among instances often reflect dependence among instances, and the instances are often heterogeneous instead of homogeneous. The assumption of independence is one of the most deeply buried assumptions in machine learning techniques and we need to fully understand the effects of such dependencies not only on relational model-building processes, but also on evaluation of these techniques.

To a large extent, current research in relational knowledge discovery is focused on learning from the data and the structure of the relations. This is an important endeavor

[1] For additional details on PROXIMITY, see <http://kdl.cs.umass.edu>.

and it deserves our attention, but there is another aspect to the discovery process for which there is little support in current technologies – data transformation. Data selection and transformation are essential to the successful application of knowledge discovery algorithms. The standardization of propositional data has limited our view of data transformation in the past, but as we move to relational data representations, this issue should return as a consideration.

1.1 Lessons Learned

The aim of this paper is to focus attention on the lessons that we have learned from analyzing a number of relational data sets. These lessons include both key new technical ideas, some of which are yet to be fully explored, and system design choices that we have found to be helpful in our analyses. First, we discuss representation and argue for the utility of a simple graph representation. Second, we characterize the space of relational features and outline the vast number of potential features that are available to relational learners. Third, we define relational autocorrelation, a characteristic of relational data that we have found to be ubiquitous in relational data. Fourth, we argue that relational data mining systems should be designed to support transformative learning in order to co-evolve knowledge and representation. Finally, we close with a short discussion of statistical issues resulting from the lack of independence in relational data that should influence algorithm design, and we point to papers treating these issues in greater detail.

The lessons contained in this paper are relevant to researchers working with both relational and propositional data. Many of the data sets considered to be i.i.d. are in fact flattened relational datasets. For example, medical records are often used to build models to predict disease given the unique symptoms of the patient. These records might include attributes that indicate the blood type of the patient's parents as well as the family history of the particular disease. These attributes are examples of information that may be better represented explicitly as relations. It is also possible that many of the patients live in the same area or work in the same building, and that this in fact determines their illness shared cause. In this case, the records are not independent even though they are represented propositionally. Researchers working with propositional data should be aware of problems that may be present if statistical dependencies inadvertently exist in their datasets. The design principles and representation issues discussed in this paper are also intended for a general audience. The intimate connection between the process of discovery and data transformation should be reflected not only in approaches to data collection but also in methods for data storage and access.

Although we will use the 2001 KDD Cup gene data (described in the next section) to provide specific examples throughout this paper, the lessons have been garnered through work in many different relational domains, including fraud detection, citation analysis, financial and corporate data, as well as movie data.

2. 2001 KDD Cup

The examples we provide are drawn for our experience participating in the 2001 KDD Cup (Cheng et al. 2002). For the past 6 years, the KDD Cup has been held in conjunction with the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. One specific goal of the competition is to provide the KDD community with a common laboratory where current research can be evaluated on practical problems. Last year was the first time that a relational data set was included in a challenge problem, indicating an increase in both the awareness of, and the need for, focused research efforts in relational discovery systems.

Rapid advances in genome mapping have increased the interest in mining data from biological domains; consequently, the data for the competition were drawn from this domain and in particular, the relational tasks were taken from the field of functional genomics. The competition was composed of three classification tasks on biological datasets; two of these tasks involved a dataset containing information about the yeast genome at both the gene level and at the protein level. Genes code for proteins and these proteins localize in various parts of the cell and interact with each other to perform various functions. For simplicity, the rest of this paper will refer to 'genes' only and treat gene as synonymous with protein even though information in the data refer to both proteins and genes.

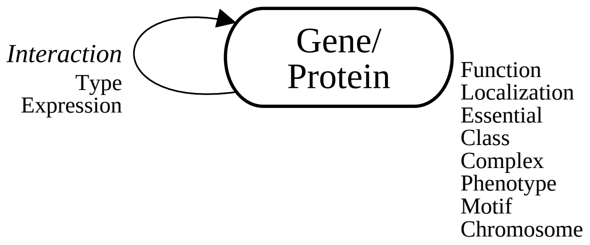


Figure 1: Gene data schema.

Figure 1 shows the data schema for the gene data from the 2001 KDD Cup competition, as represented in PROXIMITY. The data contain information about 1243 genes from the yeast genome with 1734 symmetric interaction links. The average number of interactions per gene is 2.6 (min=0,max=20). The training set consisted of 862 genes; class label information was withheld for 381 randomly selected genes in the test set. The tasks were to predict function and localization for each of the genes in the test set. There are 15 functional categories and each gene can be associated with more than one function. The average number of functions per gene is 2.6

(min=1,max=6). There are 15 locations and each gene localizes in a single location. There are six additional attributes intrinsic to genes and two attributes concerning the interactions between genes. These data illustrate the need for representing attribute information on links. Type is a discrete attribute characterizing the gene interactions. Expression is a continuous attribute in the range [-1,1] measuring the strength of the interaction between the genes.

Tasks 2 & 3 of the competition were to predict function and localization of these proteins; we placed 12 out of 41 for function prediction and 10 out of 45 for localization prediction. We used PROXIMITY to construct 124 relational features in the gene data, and then built Simple Bayesian classifiers with these features. Our test set accuracy for predicting function was 92.6% (1st place: 93.6%), for localization it was 66.14% (1st place: 72%).

3. Data Representation

3.1 Relational Representation

A common approach to learning from relational data is to "propositionalize" the data rather than retain its inherent structure. A great deal of research has been conducted on machine learning techniques for propositional data, and it is often possible to build accurate models of relational data by flattening the data and applying these algorithms. Flattening refers to the process of making each instance identical by either duplicating or aggregating the relational information. For example, the gene data could be flattened into identical instances by taking the average/modal attribute values of a gene's linked genes and appending these new aggregated attributes to the list of intrinsic attribute for each gene. However, flattening relational data removes the richer relational structure and in doing so, may impair learning. In addition, flattening has many other associated problems, which could result in incorrect statistical inferences and/or impaired learning.

Specifically, flattening relational data can result in a combinatorial explosion in either the number of instances or the number of attributes, depending upon whether one decides to duplicate or aggregate. However, a more serious issue lies in the fact that both duplicating and aggregating have the potential to produce biased parameter estimates (Jensen 1998). Flattening destroys any record of the relationships among instances; because the flattened data cannot account for dependencies in the original dataset, estimates of statistical significance on flattened data may be severely biased. Removing the relationships in the data also isolates instances from information about the predictions made on other objects. This inferential isolation eliminates the possibility of using predictions about related instances to inform inferences about other instances. Figure 2 shows that genes cluster by both location and function. Flattening the gene data would preclude the application of an iterative classification technique (Neville and Jensen 2000) that uses predictions made with high confidence to improve predictions in later iterations or other approaches that use relational structure to improve inference (e..g, Friedman, Getoor, Koller, and Pfeffer 1999). Flattening may also result in a representational mismatch of knowledge. Many simple relational concepts are extremely complex to represent in propositional form, yet effective learning requires that exactly the right attributes be identified up front.

One of the main drawbacks of a flattened representation is loss of information. The lost information cannot be reconstructed if it is later determined useful or necessary for the discovery process. However, propositionalization of relational data for learning is often a good approach used by many in the community, including ourselves. In fact, both winning teams in the 2001 KDD Cup used this approach. Maintaining a relational representation of the data does not prohibit flattening dynamically when necessary to learn models and allows for greater flexibility as research progresses in relational model building techniques.

3.2 Simple Graph Representation

We advocate using a simple graph structure to represent relational data. Several nearly equivalent formalisms exist for representing relational data sets including graphs, database tables and first-order logic statements. However, graph representations facilitate reasoning about networks of objects and support the flexible schemas described in the next section.

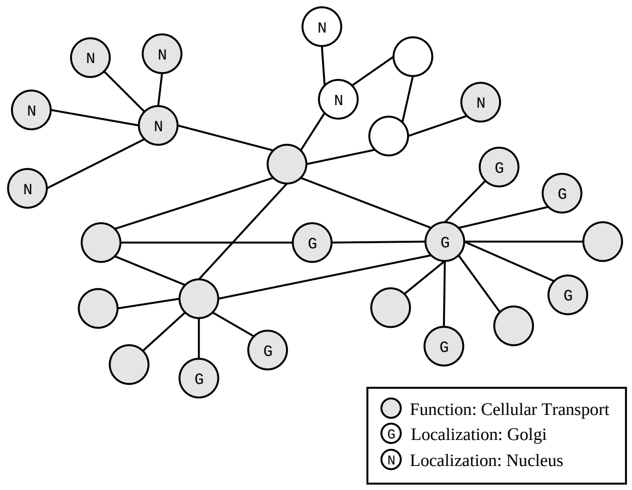


Figure 2: Gene data fragment.

PROXIMITY represents data using objects, links and attributes. Objects are used primarily to represent people, places, and things. Links represent relationships between objects, such as 'mother-of' or 'made-in' or 'part-of.' Attributes represent basic characteristics of items, either objects or links. Attributes have a name (e.g. 'City') and values (e.g. 'Amherst'). Attributes may be associated with objects and/or links and it is possible for many attribute values to be associated with any one item. In addition to these basic data structures, PROXIMITY can also represent views of the graph as collections of subgraphs. Figure 2 contains a fragment of the yeast gene data from the 2001 KDD Cup as an example. Nodes in the graphs represent genes, and links represent interactions between the proteins that the genes produce.

3.3 Flexible Schema

The PROXIMITY data schema is much more flexible than those commonly used in relational databases. A traditional database schema assumes homogenous object types, storing records for each type of object in a separate table with a fixed numbers of attributes. Once the schema is specified it can be difficult to change. It can also be difficult to change the type of an individual object, or insert records for objects whose type is uncertain. Iterative structuring or "sense-making" activities that are central to knowledge discovery are not easily supported by these traditional fixed schemas.

In contrast, PROXIMITY's schema facilitates a flexible approach to data representation. It allows for the transformation of the data as necessary throughout the discovery process. Instead of associating a fixed set of attributes with objects of a given type, attributes are each stored in a separate table. It is no longer assumed that any set of objects have the same structure, each object can be associated with any set of attributes. This approach escapes the rigid record typing of a traditional schema, and it enables analysts to introduce and transform data structures as analysis progresses.

Specifically, PROXIMITY's flexible schemas support a adaptable type system, facilitate attribute creation and allow for efficient scaling. The *type* of an object or link is an attribute like any other. An object or link can have no type, one type, or many types, and types of objects and links can be easily changed. New attributes can be added easily, without requiring the attribute be added to every object of a given type, even when values are not known. This type of representation also allows for fast access to groups of objects/links instead of fast access to full attribute information for a given object or relation. This tradeoff of row-view for column-view allows for fast views of collections of heterogeneous subgraphs.

4. Use Many Types of Relational Features

In propositional datasets with sparse information, it can be hard to build accurate models. However, relational data may contain a wealth of information in the relationships even if there is only sparse attribute information for the objects. In many transactional datasets such as financial transactions or telephone call detail, the bulk of the data are contained in relations. In these situations little is known about the objects in isolation but it is still possible to build models using the transaction patterns. Fortunately, the relational structure provides a wealth of opportunities for construction of new features to use in building models.

4.1 Relational Features

We define *feature* as mapping between raw data and a low-level inference. For example, a feature in the gene data might be *Function=cellular organization*. In this case, the feature combines an attribute (function), an operator and a value. Typically, many features are combined into a higher-level model such as a decision tree or a rule set. Relational features are used by models that predict the value of an attribute based on the attributes of related objects. For example, we might use the localization of gene *y* to predict the function of gene *x* if *x* and *y* interact together. Relational features are similar to the features described above in that they identify both an attribute and a way of testing the values of the attribute. However, relation features may also identify a particular relation (e.g. *Interaction(x,y)*) that links a single object *x* to a set of related objects *Y*. If this is the case, the attribute referenced by the feature may belong to the related objects *Y* and the test is conducted on the set of attribute values in the objects in *Y*. For example, the relational feature:

$$\begin{gathered} Mode(Localization(Y)) = nucleus \\ \quad where\ Gene(x), Y = \{y \mid Interaction(x,y)\} \end{gathered}$$

determines whether the most prevalent localization of all the genes interacting with *x* is equal to *nucleus*.

When the relation between *x* and *Y* is one-to-many, a relational feature must considers a set of attribute values on the objects *Y*. In this situation, standard database aggregation functions (e.g. *max*, *mode*, *average*) can be used to map sets of values into single values. For example, if a gene' s location depends on the chromosome values of linked genes, and a gene interacts with five other genes, then a model could aggregate the five values of chromosome with a function such as *mode*. An alternative approach is to use first-order features that produce boolean values characterizing the neighborhood. For example, a first-order feature might determine whether another gene with a particular function and location interacted with a given gene. The attribute would be true if one or more genes met the function and location criteria.

4.2 Relational Feature Space

The size of the potential feature space for relational data is enormous, as is evidenced by the search space considered in many ILP systems. Flach and Lavrac have outlined a framework for first-order features (2000) but the treatment is limited to features consisting of conjunctions of literals. Much of the work in relational learning outside the ILP community has considered features using the aggregation functions described in the previous section (e.g., Friedman et. al. 1999). Our own work for the KDD Cup consisted of creating over a hundred aggregated relational features. However, beyond aggregation of attribute values, there is a wealth of potential features concerning the structure of the data itself. Figure 3 outlines a range of feature types made possible by relational data. The x-axis represents the scope of the feature. *Individual* refers to features about objects in isolation. *Local* refers to features constructed from the local relational neighborhood on objects. *Global* refers to features constructed from the entire graph of objects. Individual features can only use attributes because an object in isolation has no graph structure. However, both local and global features can consider either graph structure or item attributes, or both.

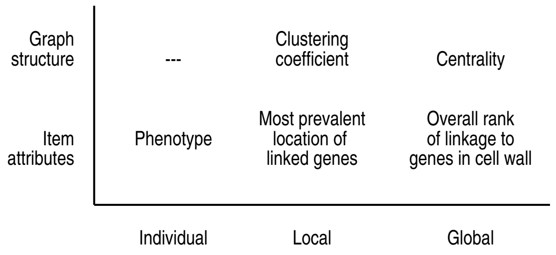


Figure 3: Feature space graphic

Figure 3 presents examples of features in each of the five types. *Phenotype* is an intrinsic attribute of genes that is provided in the raw data. Clustering coefficient is defined as the ratio of the number of linked neighbors (those who also link to each other) to the possible number of linked neighbors (Watts 1999). This measures the connectedness of surrounding genes. *Most-prevalent-location* was described in section 4.1; it considers the most prevalent localization of all the genes interacting with a given gene. Centrality measures help to determine the relative importance of nodes in the graph (Wassermann and Faust 1994). Betweenness centrality counts the number of shortest paths in the graph that travel through each gene. *Overall-rank-of-cell-wall-linkage* is a two-step feature. First the degree for each gene is measured with respect to genes located in the cell wall. Then all the genes receive a ranked according to these degrees. The highest ranked gene is the one that interacts with more genes located in the cell wall than any other gene in the genome.

Most, if not all of the features considered by current relational data mining systems fall into the categories of *Individual* and *Local Item-Attributes*. Structural features such as clustering coefficient and centrality are more common in social network analysis, and other algorithms such as Kleinberg’s hubs and authorities algorithm (Kleinberg 1999) that have been developed in the computer science community have only begun to be explored. Kleinberg' s algorithm was developed to quantify two measures of web page “interestingness.” Hubs and authorities are defined recursively – a web page is an authority if it is linked to by many hubs, and it is a hub if it provides links to many authorities. The algorithm iteratively calculates hub and authority weights on all pages simultaneously using the link structure. Features such as these require a global view of the graph to be calculated. Features of this type have yet to be exploited by the community, and are available only if you retain a relational representation. Much work has been done in traditional machine learning to determine the utility of features and select the most useful. Both this work and work in automatic feature construction, are important to relational knowledge discovery systems because the number of potential relational features.

5. Autocorrelation is Ubiquitous

Our analysis indicates that *relational autocorrelation* is a common characteristic of relational data[2]. Informally, relational autocorrelation occurs when the values of a given attribute are highly uniform among objects that share a common neighbor. The fragment of the gene data in Figure 2 shows how many linked genes have highly correlated functions and locations.

We will define relational correlation *C(X,f,P,Y,g)* with respect to two sets of objects *X* and *Y*, two attributes *f* and *g* on objects in *X* and *Y*, respectively, and a set of paths *P* that connect objects in *X* and *Y*.

Definition: *Relational correlation C* is the correlation between all pairs *(f(x),g(y))* linked by paths in *P*. ■

Given the pairs of values that these elements define, traditional measures such as information gain, chi-square, and Pearson' s contingency coefficient can be used to assess the correlation between values of the attributes *f* and *g* on objects connected by paths in *P*. The range of *C* depends on the measure of correlation used.

We can use the definition of relational correlation *C(X,f,P,Y,g)* to define relational autocorrelation as the correlation between the same attribute on distinct objects belonging to the same set.

Definition: *Relational autocorrelation C' is:*

$$C'(X,f,P) \equiv C(X,f,P,X,f) \text{ for } P = \{p(x_i,x_j) \mid x_i \neq x_j\} \quad ■$$

For example, *C'* could be defined with respect to gene objects, the attribute *localization* on genes, and paths formed by traversing *Interaction* links that connect the genes to other genes.

5.1 Autocorrelation in Relational Datasets

If the underlying measure of correlation varies between zero and one, then *C'=1* indicates that the value of the

[2] Much of this discussion is drawn from earlier papers (Jensen and Neville 2002a,b).

attribute for a specific node x_i is always equal to all other nodes x_j reachable by a path in P. When $C'=0$, values of $f(X)$ are independent. Table 1 gives estimates of relational autocorrelation for gene function and localization, linked through other genes. For a measure of correlation, Table 3 uses Pearson's corrected contingency coefficient (Sachs 1992), a measure that produces an easily interpreted value between zero and one. Autocorrelation is fairly strong for both class labels.

Table 1: Autocorrelation in the gene data

Autocorrelation Type	Value
C'(Function,Gene,Interaction)	0.52
C'(Localization,Gene,Interaction)	0.76

In addition to the gene data, we have encountered many other examples of high relational autocorrelation. For example, in our study of publicly traded companies, we found that when persons served as officers or directors of multiple companies, the companies were often in the same industry. Similarly, in data from the Internet Movie Database, movies linked through a common studio, director or producer have highly autocorrelated box-office receipts. Such autocorrelation has been identified in other domains as well. For example, fraud in mobile phone networks has been found to be highly autocorrelated (Cortes et al. 2001). The topics of authoritative web pages are highly autocorrelated when linked through directory pages that serve as "hubs" (Kleinberg 1999). Similarly, the topics of articles in the scientific literature are often highly autocorrelated when linked through review articles.

Relational autocorrelation represents an extremely important type of knowledge about relational data, one that is just beginning to be explored and exploited for learning statistical models of relational data (Neville and Jensen 2000; Slattery and Mitchell 2000). Deterministic models representing the extreme form of relational autocorrelation have been learned for years by ILP systems. By representing and using relational autocorrelation, statistical models can make use of both partially labeled data sets and high-confidence inferences about the class labels of some nodes to increase the confidence with which inferences can be made about nearby nodes.

Relational autocorrelation can also complicate learning and evaluating relational models. These issues are outlined further in section 7.

6. Transformative learning

Successful knowledge discovery is often an iterative process during which analysts coevolve their domain knowledge and the way in which they represent data about that domain. Building models of raw data is frequently unsuccessful but this phase (as well as subsequent phases) can suggest new ways to view the data, either by modification or by constructions of new features. Knowledge discovery can also be approached as a hierarchical process where simpler concepts need to be learned first and then used together to model more complex concepts. Supporting an iterative approach to discovery is useful for systems designed to analyze propositional data, and it is vital for systems designed to analyze relational data.

6.1 Supportive Architecture

We have found that the discovery process is intimately connected to how data are represented. Sometimes it is hard to know the 'right' way to represent relational data for analysis. For example, should a financial transaction between two people be represented as a 'transaction' object with relations to the two people or just as a relation between the two people? It could depend on the task. If the user is trying to model fraudulent businesses, then the first representation might be more useful to look at all transactions conducted at a particular site. However, if the user is trying to model fraudulent people, the second representation might be more useful to view aggregate behavior of an individual. Often the task is only to model 'fraud' and it is during the process of discovery that one or the other view becomes the focus of the analysis. In this situation, a hard and fast choice of representation up front could not only slow down discovery but also prevent it completely.

We have designed PROXIMITY with an architecture suited to managing structured data and methods that can both support and direct effective data transformations. In addition to the simple graph representation described earlier, PROXIMITY provides views of the graph using collections of subgraphs. For example, in the gene data we could create a collection of all the genes that localize in the cell wall and then analyze the properties of these genes to try to predict function. This would be useful if we were able to build an accurate model of localization that could be applied to test data first and then have a second model that predicts function given localizations.

We could also create a collection of subgraphs in the gene data to view each object in its local relational neighborhood. In this case, each subgraph in the collection would consist of a core gene object and there would be a separate subgraph for each gene in the data. Each subgraph could contain all the other genes that the core gene interacts with; these genes may be duplicated across subgraphs. This approach allows for fast calculation of features concerning the local neighborhood of a gene. Most of the attributes we constructed for the KDD Cup were calculated using such a view of the gene data. The view made it simple to calculate such attributes as 'count of linked genes in golgi' and 'sum of interaction expressions to linked genes whose function is cellular transport.'

Views of the graph facilitate the analysis process by allowing an analyst to filter and abstract the data. Filtering the data is often useful in data sets with large numbers of instances or attributes, but it can also be useful when the data have many types of objects or links as well. Often, filter the data can reveal new associations that were previously hidden or swamped out by other portions of the data. Our analysis of the gene data included calculating autocorrelations for genes linked in various ways. For, example we filtered the links by type and expression. We also looked at pairs of interacting genes that shared the same function or location. Table 2 reports several of the high autocorrelations we found using filtered links.

Table 2: Autocorrelation in the gene data

Autocorrelation Type	Value
C'(Func=transcription,Gene,Interact-same-loc)	0.92
C'(Func=transport,Gene,Interact-type=gen-phys)	0.96
C'(Loc=nucleus,Gene,Interact-type=genetic)	0.84
C'(Loc=golgi,Gene,Interact-same-func)	0.62

6.2 Inductive Closure

The ability to transform the data easily and repeatedly throughout the discovery process is important not only to improve efficiency but it is also crucial to evolving knowledge about the domain. To this end, we designed a system with *inductive closure*. This denotes that the system is closed under induction -- the results of any analysis feed directly back into the system to be reused by further analysis. This design choice is again motivated by the iterative nature of the discovery process and a view of the database as a blackboard. If relationships in the data are to be used effectively to improve the quality of predictions, a system's model output should feed directly into the next model-building phase.

PROXIMITY modules use a small number of special purpose data structures to make interaction between modules possible and is accompanied with a scripting language to support dynamic compositions by analysts using any number of analysis methods. PROXIMITY classifiers operate on collections, use attributes and produce attributes to record predictions. PROXIMITY graph queries take a subgraph specification, output collections of matches and facilitate adding new object/links and or calculating attributes based on the information in the subgraphs. PROXIMITY graph calculation modules that construct views such as connected components or paths operate on collections and produce either collections or attributes.

7. Statistical Opportunities and Challenges

Relational data have the potential to drastically improve not only the accuracy of learned models, but also the quality of discovered patterns. However, the dependence among instances in the data introduces new statistical challenges and opportunities for relational learning algorithms. As we seek to represent and use relational autocorrelation in statistical models of relational data, we will need to adjust for its effects on feature selection and evaluation. Solving these issues could result in huge benefits by reducing the sample complexity of the learning algorithms and increasing the accuracy of the learned models.

7.1 Feature Selection Bias

Two common characteristics of relational data sets — *concentrated linkage* and *relational autocorrelation* — can cause learning algorithms to be strongly biased toward certain features, irrespective of their predictive power. In a related paper (Jensen and Neville 2002a) we show how dependence among the values of a class label in relational data can complicate feature selection in methods for machine learning. We show how linkage and autocorrelation can combine to reduce the *effective sample size* of some data sets, introduce additional variance, and lead to feature selection bias. To our knowledge, no current relational learning algorithm accounts for this bias. Resampling can be used to obtain accurate estimates of variance for each feature and we are currently investigating techniques to use those estimates to improve feature selection in relational data.

7.2 Bias in Evaluation

Linkage and autocorrelation can also cause traditional methods of evaluation to greatly overestimate the accuracy of induced models on test sets. Accurate evaluation of learning algorithms is central to successful research in relational learning. The most common method for evaluating a learning algorithm is to partition a given data sample into training and test sets, construct a model on the training set and evaluate the accuracy of that model on the test set. Dependence among the values of a class label in relational data can cause strong biases in the estimated accuracy of learned models when accuracy is measured in this way. In general, current techniques for evaluating relational learning algorithms do not account for this bias. We've presented a new sampling algorithm that can be applied to any relational data set to eliminate this bias in a paper submitted elsewhere (Jensen and Neville 2002b). This approach removes the dependence between training and test sets by confining any correlation among class labels to a single subsample (i.e. within the subsamples instead of across the subsamples).

7.3 Improving Predictions

In addition, maintaining relational representations allows inference procedures to exploit relational autocorrelation to improve the predictive accuracy of models. By preserving the relational structure of the data, we can exploit the connections between objects to improve classification accuracy. We have developed an *iterative classification* technique (Neville and Jensen 2000) based on the premise that if two target objects are related, inferring the class of one can tell us something about the class of the other. As the name suggests, iterative classification runs a classifier many times on the same collection of subgraphs, recalculating attribute values after each iteration. Inferences made with high confidence in initial iterations of the classifier are fed back into the data to strengthen inferences about related objects in subsequent iterations.

8. Conclusions

The biases associated with linkage and autocorrelation indicate the importance of maintaining relational data representations, rather than propositionalizing data. Maintaining a relational data representation makes it possible to assess the statistical effects of linkage and autocorrelation, and to adjust for the resulting bias. In addition, relational representations allow inference procedures to exploit relational autocorrelation to improve the predictive accuracy of models. Finally, relational representations extend the space of potential relational features; in particular, global and structural features present a set of previously unexplored features.

A co-evolutionary approach to knowledge formation and data transformation addresses the “chicken and egg” character of knowledge discovery. Knowing the “right” representation can be crucial to the discovery process but often this knowledge is not known up front and needs to be learned during analysis. The additional information contained in relationships allows for alternate views of the data and as such contain many possibilities for transformation during learning. Systems designed to support an iterative discovery approach promise to clarify the utility of various representations and will enable dramatic improvements in knowledge discovery processes.

Acknowledgements

This paper would not have been possible without the collaborative environment within the Knowledge Discovery Laboratory. In particular, the authors thank Matt Cornell and Hannah Blau for their contributions to the PROXIMITY project over the past three years. This research is supported under a National Science Foundation Graduate Research Fellowship and by DARPA and AFRL, AFMC, USAF under contract numbers F30602-00-2-0597 and F30602-01-2-0566. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements either expressed or implied, of the DARPA, the AFRL or the U.S. Government.

References

Cheng, J., C. Hatzis, H. Hayashi, M. Krogel, S. Morishita, D. Page, J. Sese. (2002). KDD Cup 2001 Report. SIGKDD Explorations.

Cortes, C., D. Pregibon, and C. Volinsky (2001). Communities of Interest. Proceedings Intelligent Data Analysis 2001.

Dzeroski, S. and N. Lavrac, (Eds.) (2001). Relational Data Mining. Berlin: Springer.

Flach, P. and N. Lavrac. (2000). The role of feature construction in inductive rule learning. Proc. of ICML2000 Workshop on Attribute-value and Relational Learning.

Friedman, N., L. Getoor, D. Koller, and A. Pfeffer (1999). Learning Probabilistic Relational Models. In *IJCAI'99*. 1300-1309.

Jensen, D. (1998). Statistical Challenges to Inductive Inference in Linked Data. Proceedings of the 7th International Workshop on Artificial Intelligence and Statistics.

Jensen, D. and J. Neville (2002a). Linkage and Autocorrelation cause bias in relational feature selection. Proc. 19th International Conference on Machine Learning.

Jensen, D. and J. Neville (2002b). Autocorrelation and linkage cause bias in evaluation of relational learners. Submitted: 12th International Conference on ILP.

Kleinberg, J. (1999). Authoritative sources in a hyperlinked environment. Journal of the ACM 46:604-632,

Neville, J. and D. Jensen (2000). Iterative Classification in Relational Data. AAAI Workshop on Learning Statistical Models from Relational Data, 42-49.

Sachs, L. (1982). Applied Statistics. Springer-Verlag.

Silverstein, G. and Pazzani, M. (1991). Relational cliches: Constraining constructive induction during relational learning. Proceedings of the Eighth International Workshop on Machine Learning.

Slattery, S., and Mitchell, T. (2000). Discovering test set regularities in relational domains. Seventeenth International Conference on Machine Learning.

Wassermann, S. and Faust, K (1994). Social Network Analysis: Methods and Applications, Cambridge: Cambridge University Press.

Watts, D. (1999). Small Worlds. Princeton Univ. Press.

Exploiting Relational Structure to Understand Publication Patterns in High-Energy Physics

Amy McGovern, Lisa Friedland, Michael Hay, Brian Gallagher, Andrew Fast,
Jennifer Neville, David Jensen
Knowledge Discovery Laboratory
University of Massachusetts Amherst
140 Governors Drive, Amherst, MA 01003
amy,lfriedl,mhay,bgallag,afast,jneville,jensen@cs.umass.edu

ABSTRACT

We analyze publication patterns in theoretical high-energy physics using a relational learning approach. We focus our analyses on four related areas: understanding and identifying patterns of citations, examining publication patterns at the author level, predicting whether a paper will be accepted by specific journals, and identifying research communities from the citation patterns and paper text. Each of these analyses contributes to an overall understanding of theoretical high-energy physics that could not have been achieved without examining each area in detail.

1. INTRODUCTION

We identify interesting patterns and relationships in the theoretical high-energy physics publishing community using a relational learning approach. We focus on several high-level questions:

- Can we predict why some papers receive more citations than others? What are the trends in citations and references?
- What factors contribute to an author's influence? Can we identify measures of influence? Can we predict potential award winners?
- What factors contribute to journal publication? Can we predict whether a paper will appear in a particular journal?
- Can we identify schools of thought or communities in theoretical high-energy physics? Who are the most authoritative authors for each community?

We analyzed these questions using a relational approach. We constructed the relational schema shown in Figure 1. This schema provides a rich representation for the *hep-th* data and supports many interesting analysis and prediction tasks. In the following sections, we discuss our analyses and present our results, including:

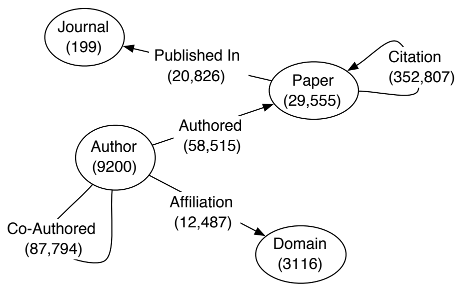


Figure 1: Schema extracted from the abstracts and citation data. Objects are represented by vertices and relations by edges; numbers in parentheses are object and relation counts.

- Approximately 26% of the people in *hep-th* wrote papers that received 80% of the citations.
- Edward Witten is the most influential author in theoretical high-energy physics.
- Papers with only a single author are less likely to be published in journals than multi-authored papers.
- Authors tend to prefer particular journals, that is, a journal's name is autocorrelated through authors.
- Authors tend to publish within topics (i.e., topics are also autocorrelated though authors).

These findings and many others are explained in more detail in sections 3 through 6.

2. DATA REPRESENTATION

We use a relational representation from the *hep-th* data. Our representation uses an attributed graph, $G = (V, E)$. Objects, such as authors, journals, and papers, are represented as vertices in the graph. Relations between these objects, such as *published-in(paper, journal)*, are represented by links between the objects. If there is a relation $r(o_1, o_2)$, then $o_1, o_2 \in V$ and $r \in E$. Attributes are associated with objects, such as *author.last-name*, or edges, such as *authored.rank*.

Figure 1 shows the objects and relations we use, along with their counts in the database. Details on the attributes and on how we extracted them from the *hep-th* data are given in Appendix A. The process of author consolidation, that is, determining if the John Smith who wrote paper 1 is the same person as the J. Smith who wrote paper 2, was greatly facilitated by the relational structure [1]. Details of our consolidation approach are in Appendix B.

3. CITATION ANALYSIS

Our first analysis focuses on the papers and citation relations between them. We start by identifying patterns and correlations in this data. We use this to analyze why some papers are more popular than others and we build a relational model to predict popular papers.

3.1 Citation Graph Analysis

The citations graph is comprised of 1,928 separate connected components. The largest contains 27,400 papers, while all the others contains 10 or fewer papers. The growth in popularity of *arXiv* and *hep-th* (1397 papers in 1992 to 3312 in 2002) and the limited time frame of the data set cause edge effects on the early and late years; so we often concentrate on the more stable middle years. Figure 2a shows these effects. We break both references and citations into self and non-self categories. A self citation or reference means that there is a shared author between the two papers. 18% of the citations in *hep-th* are self citations. On a per-paper basis, an average of 28% of a paper's references cite its authors past work and 34% of a paper's citations are from its authors. Papers with low citation and reference counts generate a large proportion of the self citations thus the per-paper averages are higher than the overall percentage. The number of non-self citations peaks for papers submitted in 1996.

Because papers are often submitted to *hep-th* before being published in a journal, we hypothesized that papers might receive citations in two peaks. In particular, a paper could be cited by other papers in *hep-th* as soon as they were submitted to *arXiv* and again after being published in a journal. Figure 2b shows the number of citations that each paper received in the years following its submission to *arXiv*. Starting with the overall mean (the thick line), we can see that papers generally receive the most citations in the year following submission to *arXiv*. Since the average time from a paper's submission to *arXiv* until it appears in a journal is about one year, this peak likely coincides with journal publication. It is interesting to note that papers receive an average of two citations in the year prior to journal publication. This demonstrates the success of *arXiv* by allowing people to cite work before it has been published.

The pattern of citations for papers submitted to *arXiv* in 1992 is also interesting. In this case, the peak is two years after submission to *arXiv*. This delay can be explained by *arXiv*'s growing popularity as the use of the Internet grew; in 1992, their audience was limited. In later years (e.g., 1995, 1999), the number of citations increases more quickly due to the larger number of authors with Internet access.

Figure 2c shows the average number of non-self citations for papers that have been published in a journal versus unpublished papers. Papers that are published in a journal have a significantly higher average non-self citation rate than papers that are published only on *arXiv*. Although many people in the high energy physics community have access to *arXiv*, it is clear that either journal publication is still important in increasing a paper's visibility, or that authors writing highly cited papers still seek journal publication.

Author authority papers	Num. in top 10	Num. in top 50	Non-self citations
Edward Witten	4	14	18716
Juan M. Maldacena	2	6	8076
Steven S. Gubser	2	4	5067
Igor R. Klebanov	1	4	5843
Leonard Susskind	1	4	5526
Joseph Polchinski	1	4	5535
Paul K. Townsend	1	3	4991
Stephen H. Shenker	1	2	2300
Michael R. Douglas	0	5	5787
Nathan Seiberg	0	3	9911
Cumrun Vafa	0	3	8594
Andrew Strominger	0	3	6480
Petr Horava	0	2	1936
Daniel Z. Freedman	0	2	1874

Table 1: Authors of the top 10 and top 50 most authoritative papers and the total number of non-self citations that these authors have received in *hep-th*.

Author of hub papers	Num. in top 1%	Num. in top 5%	Non-self references
Igor R. Klebanov	15	31	5843
Arkady A. Tseytlin	10	29	5352
Steven S. Gubser	9	28	5067
Ofer Aharony	8	19	2307
Washington Taylor IV	6	7	2115
Alberto Zaffaroni	6	13	1369
Clifford V. Johnson	6	21	1615

Table 2: Authors of the top 1 and top 5 percent hub papers and the total number of non-self references that these authors have made.

The hubs and authorities algorithm [6] can used on the citation graph to identify authoritative papers and potential review papers. A *hub* is an object that points to many authorities. This is likely to be a review paper. An *authority* is an object that is pointed to by many hubs. Once we identified the most authoritative papers, we examined the authorship for these papers. Table 1 shows the authors who have written at least two of the top 10 and top 50 most authoritative papers. As many of these names appear again when we study influential authors, we discuss their specifics in section 4. In general, the authors of these highly authoritative papers include a number of award winners including MacArthur Foundation fellows, Dirac winners, and Fields medalists. They hail from many prestigious institutions including the Institute for Advanced Studies at Princeton, Princeton University, Harvard, Rutgers, Stanford, UC Santa Barbara, Cambridge, UC Berkeley, and MIT.

Table 2 shows some authors who have written top hub papers in the database. We were interested in the question of whether some authors write mostly review papers. In *arXiv*, no author has written more than one of the top 10 or top 50 hub papers. However, if we examine the top 1% and top 5%, several authors show up as frequently writing review papers. The top three authors on this list, Klebanov, Tseytlin, and

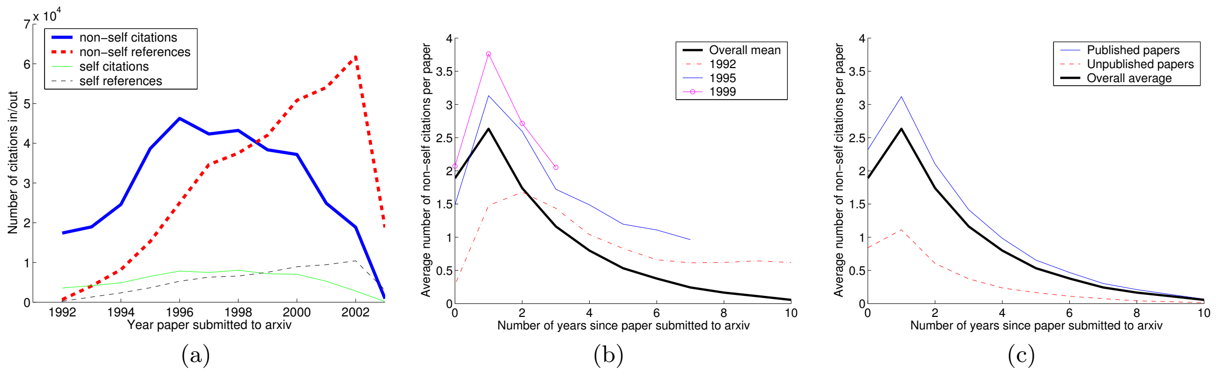


Figure 2: Temporal citation and reference patterns for papers submitted to *arXiv*. (a) Total number of non-self and self citations and references by year. (b) Citations patterns for all papers. (c) Citation patterns for published versus unpublished papers.

Attribute	**Through**	**Score**
arXiv area of paper	Author	0.72
Num. downloads first 60 days	Author	0.55
Journal name	Author	0.69
Clustered topic of paper	Author	0.54
Authority score on coauthor graph	Paper	0.74
arXiv area of cited paper	Paper	0.70
Num. of coauthors	Paper	0.45
Num. downloads first 60 days	Journal	0.42

Table 3: Selected autocorrelation scores.

Attribute 1	**Attribute 2**	**Score**
	For paper	
Authority score	Num. of citations	0.85
Area (from *arXiv*)	References (binned)	0.68
Hub score	Num. of references	0.62
Num. downloads first 60 days	Num. of citations	0.57
Is paper published	Citations (binned)	0.46
	For author	
Num. of publications	Num. of distinct coauthors	0.85
Num. of distinct coauthors	Num. of non-self citations	0.59

Table 4: Selected correlation scores between attributes.

Gubser, are frequent co-authors. Table 2 contains no major award winners and represents a slightly different list of institutions than Table 1 including Princeton, Ohio State, Rutgers, MIT, CERN and the University of Durham, UK.

3.2 Citation Data Dependencies

To better understand what makes papers popular and identify trends and patterns in the citation data, we analyzed correlations in the citation data. For discrete attributes, we used chi-square corrected contingency coefficients; for continuous attributes we used correlation coefficient [13] Tables 3 and 4 list significant correlations in the data. All reported correlations are significant at the $p < 0.0001$ level.

The number of times that a paper is downloaded is correlated with the number of non-self citations of that paper. This is not surprising as one expects more frequently downloaded papers to be cited more frequently.

In addition to correlations among variables of a single object, we also measured *autocorrelation* throughout the data graph [3]. Autocorrelation is a statistical dependency between the values of the same variable on related objects, also known as homophily [7]. For example, the number of downloads of a paper is autocorrelated through authors. This means that if one of an author's papers is frequently downloaded, other papers by the same author are likely to be downloaded as well.

3.3 Predicting Popular Papers

We used relational probability trees (RPTs) [11] for several modeling tasks. For each task, we sampled papers temporally, training the model on papers from one year and testing on the following year's papers. To avoid edge effects, we considered only papers from 1995 to 2000. For classification, the models considered characteristics of papers, their referenced papers, authors, and other past papers written by the authors. Some example attributes include the number of pages of the paper, its file size in KB, keywords, the author's number of past co-authors, the number of past publications for each author, and the number of citations received by a cited paper. Attributes were calculated for each temporal sample. For instance, to predict the class label on a paper submitted in 1997, the model considered the citation/publication history of related objects up to and including 1996.

The first modeling task involved predicting the number of non-self citations that a paper will receive. We categorized the number of non-self citations into quartiles: {0-1, 2-5, 6-14, >14}. Default classification accuracy is approximately 25%. Over 5 training/test splits, RPT models achieved and average accuracy of 44%. Although 44% is not an extremely high accuracy, it is not likely that we would achieve a high accuracy solely based on the information available in *hep-th*. Such measures as the quality of the paper are not able to be captured based on the performance of an author's past papers and this may not fully capture the situation.

One reason we chose to use RPT models is their selectivity. We can examine the features chosen by the trees and identify the most relevant features for the classification task. The RPT models identified that a paper has a probability of 0.85 of receiving more than 14 non-self citations if 1) the paper has more than 8 references 2) the authors have at least 2 past papers with more than 8 non-self citations 3) the authors have at least 25 past papers (that are at least 15 pages long) 4) at least 30% of cited work is unpublished.

4. AUTHOR ANALYSIS

The second part of our analysis focuses on the authors in theoretical high-energy physics. We start by analyzing the overall structure of the author subgraph and extend this understanding to identifying influential authors. We define several measures of influence and build a relational model to identify and predict award-winning authors. Finally, we predict potential award winners in theoretical high-energy physics.

4.1 Co-Author Graph Analysis

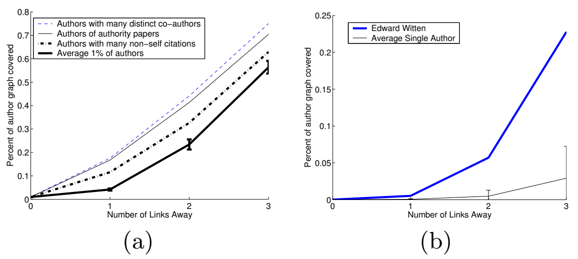


Figure 3: (a) Percent of the author graph that is one, two, and three links away from several sets of the top 1% of authors as well as from a random sampling of 1% of authors. (b) Percent of the author graph that is 1, 2, and 3 links away from Edward Witten versus the average author.

We found that the high energy physics community is tightly knit. In the graph of authors linked by co-authored relations, 7304 of the total 9200 authors belong to a single connected component. As with the paper graph, other components are all small (15 or fewer authors). When we narrowed this set of authors to authors who wrote the top 1%, 5% and 10% of the authoritative papers, we found that in each case the vast majority of the authors remained connected, with only a very small percentage in separate components. This provides evidence for the idea that influential scientists train the up-and-coming influential scientists in their labs, either as students or post-doctoral fellows [5], and co-author with them.

We also found that authors who are highly cited or have many distinct co-authors are more central to the author graph than randomly selected authors. Figure 3 shows the percentage of authors who are 1, 2, and 3 links away from authors who wrote the top 1% of authority papers, authors who have received the top 1% of non-self citations and the top 1% of authors who have co-authored with different people. These numbers are compared to 10 random samplings of 1% of the authors. Each of these sets of influential authors reaches a higher percentage of authors by following even just one co-authored relation than random. This trend continues for paths of length two or three. We also show the average degree of separation from Edward Witten, who consistently shows up as the most influential author in *hep-th*.

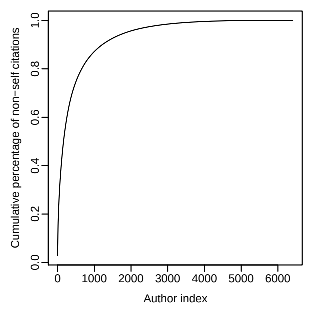


Figure 4: Cumulative percent of non-self citations received per author.

Before building a quantitative measure of an author's influence, we examined the data for general trends. From 1995 through 2000, a relatively stable window for the data set, 6405 authors submitted papers to *arXiv*. Of these authors, on average each wrote 5 papers; the median was 2. Sergei Odintsov (with 92 papers) and H. Lu and C.N. Pope (each with 84) topped the distribution. As seen in Table 6, the top authors produce high numbers of papers by co-authoring widely and frequently. The average number of distinct co-authors is 5.5. Of the papers submitted to *arXiv* in this period, each author published an average of 4 papers in journals. On their combined papers, authors recieved an average of 76 non-self citations, with a much lower median of 7. The top 10% of authors averaged 140 non-self citations.

The 80/20 rule or Pareto's Principle states that, in power law distributions, 80% of the mass is generally due to only 20% of the values (whether in science or other domains)[12, 8]. We investigated this rule in theoretical high-energy physics by examining the number of non-self citations received on a paper and author basis. In the *hep-th* data, 80% of the non-self-citations go to 17.8% of the papers and 26.3% of the authors wrote these papers. The full distribution for authors is shown graphically in Figure 4.

4.2 Author Data Dependencies

Trends and dependencies for authors are summarized in Tables 3 and 4. The number of an author's publications is correlated with the number of citations that the author receives. This means that either authors who have more citations publish more frequently or that people who publish more papers receive more citations. Perhaps more surprising is that the number of publications that an author has is correlated with the number of distinct co-authors that the author has published with. This indicates that frequently published authors do not tend to work repeatedly with only the same set of co-authors but continue to expand their research to working with new people.

We expected that authors who write authoritative papers are likely to write other authoritative papers but this was not the case. A paper's authority score was not autocorrelated through author which means that most authors will write only a few authoritative papers in their lifetime.

Information about the research styles of authors can be gained from autocorrelation scores. For instance, the number of distinct coauthors is autocorrelated through papers. That is, if you publish with other authors who publish with

1. Number of non-self citations received
2. Total number of citations received
3. Number of papers written
4. Number of papers published in journals
5. Number of papers with over 12 citations
6. Number of co-authorships
7. Number of distinct co-authors
8. Average non-self citations per paper
9. Maximum non-self citations received on any paper
10. Percentage of papers published
11. Percentage of papers with over 12 citations
12. Weighted combination of 1, 4, 5, and 9.

Table 5: Measures of author influence

many distinct people you are also likely to publish with many distinct people. Within the *arXiv* data, an author who publishes a paper in a particular journal is likely to publish his other papers in that journal as well.

4.3 Analyzing Author Influence

After gaining a general understanding of author publication patterns, we hypothesized that author influence, that is, overall reputation and impact, could be defined using the measures shown in Table 5.

(a) Overall co-authorships

Author	Count
C.N. Pope	337
H. Lu	325
S.D. Odintsov	296
Sergio Ferrara	233
Mirjam Cvetic	231

(b) Distinct co-authorships

Author	Count
Cumrun Vafa	63
Gary W. Gibbons	60
Jan de Boer	56
Sergio Ferrara	55
Antoine Van Proeyen	55

Table 6: (a) Authors who frequently co-author on papers (including repeatedly co-authoring with the same person). (b) Authors who frequently co-author with different people on papers.

We ranked the authors who submitted papers to *arXiv* from 1995 to 2000 according to each of these measures and evaluated each measure according to the number of award winning authors it ranked highly. We identified 55 winners of prestigious awards, including Nobel prize winners, MacArthur Foundation fellows, Dirac fellows, Guggenheim recipients, Fields medal winners, and Alfred P. Sloan Foundation winners. Based on the number of award winners listed in the top 100 of each ranking, we found that most of the above measures performed about equally, finding around 10 award winners. Measures 1 and 2 did best, with 14 winners. We therefore chose measure 1 to be our canonical influence measure, noting that the raw total of citations is also the one used by popular research tools[1]. Figure 7 shows the top authors and their citation counts. Heading the list, Edward Witten is a MacArthur Foundation fellow, a Fields medalist, and a Dirac fellow. Juan Maldacena, also a MacArthur Foundation fellow, is a younger researcher and looks quite likely to become the most cited author as he continues his research. This table also includes a number of other award winners.

Surprisingly, measures 10 and 11, which we constructed to indicate an author's consistency of success, performed

[1] Citeseer: http://citeseer.nj.nec.com/mostcited.html and ISI Essential Science Indicators: http://www.in-cites.com

Author	Non-self citations	# papers
Edward Witten	13806	59
Juan M. Maldacena	7334	39
Cumrun Vafa	6578	55
Nathan Seiberg	6258	45
Andrew Strominger	5371	44
Michael R. Douglas	5089	24
Igor R. Klebanov	5063	51
Joseph Polchinski	4815	25
Steven S. Gubser	4812	31
Ashoke Sen	4201	51

Table 7: Top-cited authors, based on papers 1995-2000

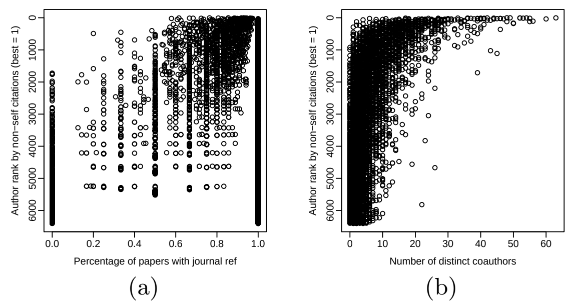


Figure 5: (a) Author influence vs. percent of papers published. (b) Author influence vs. distinct co-authors

poorly on our validation task, identifying 2 or fewer winners. Closer inspection shows that perfectionism is not the key to success. The percentage of papers published in journals varied widely among award-winners, from 100% to 0%, although the top 50% of influential authors did have a higher rate (88%) of acceptance than the bottom half (67%). This is shown graphically in Figure 5a. Percentage of papers highly cited was better correlated with non-self citations (see Figure 5b), but the measure performed poorly because it placed authors with one of one paper highly cited above those with 19 of 20 papers highly cited.

Figure 5b examines the correlation between citation count and number of coauthors. As pointed to earlier, authors with high citation counts write both frequently and widely. Even in the middle of the scale, collaborating with 10-15 other people is typical. However, anyone with over 30 co-authors is almost certain to be in the top 10%; presumably one must be extremely well-regarded to be in that kind of demand by collaborators. It is possible to have few co-authors and still receive very high citation counts. In the top 10% by non-self citation count, no one writes alone, and of the top 100 authors, only one (Donam Youm) has fewer than 10 distinct coauthors. Table 6 displays the authors with the highest co-author counts.

We wondered if a different combination of features could identify a better measure of what differentiates award-winners from other authors. To do this, we built an RPT using the set of 55 award winners and a random sample of 55 non-award winners. We performed 10-fold cross validation and achieved an average accuracy of 78% with an area under the ROC curve of 0.75. The tree chosen most frequently is shown in Figure 6.

The first split in the tree, the author's authority score, is

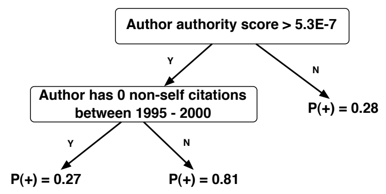

Author authority score > 5.3E-7
Y
N
Author has 0 non-self citations between 1995 - 2000
P(+) = 0.28
Y
N
P(+) = 0.27
P(+) = 0.81

Figure 6: RPT built to predict award-winning authors.

based on the authority score received when running the hubs and authorities algorithm over the undirected co-author graph.[2] This roughly correlates with authors who co-author frequently and whose co-authors also co-authored frequently.

Informed by the features in the tree as well as by our other analyses, we conjecture that some of the following highly cited authors, from the tops of the lists but relatively lacking in major awards, may soon be due for recognition: Andrew Strominger, Igor R. Klebanov, Ashoke Sen, Arkady A. Tseytlin, Paul K. Townsend, Gregory Moore, and Hirosi Ooguri.

5. PUBLICATION ANALYSIS

Influential authors are more likely to have their papers accepted by a journal. It is also clear from Figure 2 that published papers receive more citations. With this in mind, the third part of our analysis studied what other factors affected journal acceptance and used the RPT to predict journal acceptance as well as publication venue.

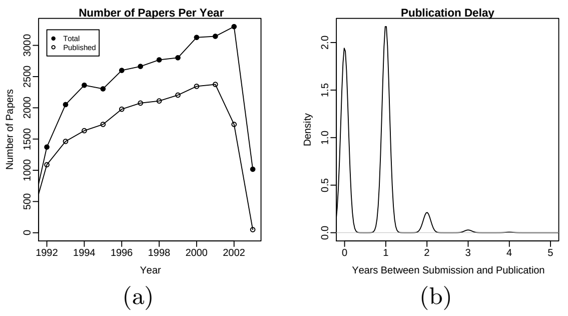


Figure 7: (a) Number of published and unpublished papers submitted to *arXiv* each year. (b) Number of years between a paper's submission to *arXiv* and it appearing in a journal.

Approximately 70% of the papers in *arXiv* have been published in a journal. Figure 7a shows the total number of papers submitted to *arXiv* each year for both published and unpublished papers. Although the total number of papers increases each year, the proportion of published and unpublished papers remains relatively constant. Figure 7b shows the distribution of the number of years between a paper's submission to *arXiv* and it appearing in a journal. Most papers, if published at all, are published within one year of submission to *arXiv*. A small number are published up to 4 years later.

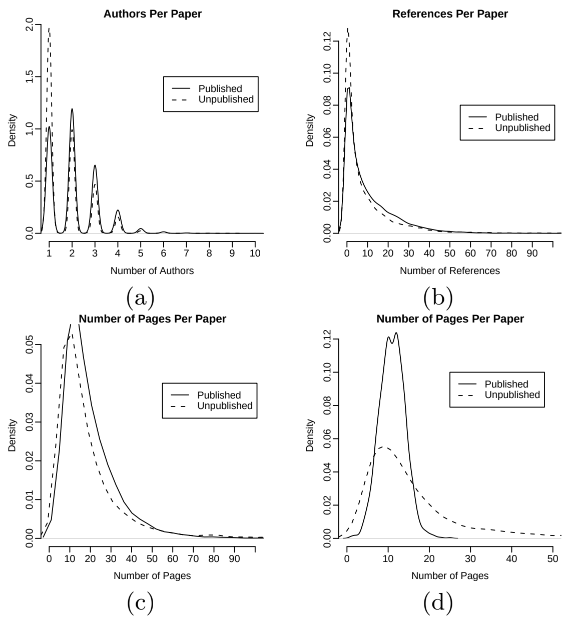


Figure 8: Characteristics that differentiate published and unpublished papers. Figures a, b, and c are from all published and unpublished papers from 1995 to 2000 inclusive. Figure d is from a sample of 3000 papers (1500 published in *Physics Letters B* and 1500 unpublished).

We analyzed the differences between the published and unpublished papers in several ways and discovered significant effects. Several of these effects are shown in Figure 8. The most surprising difference is that published papers usually have more than one author while unpublished papers are much more frequently written by a single author. This is an example of *degree disparity* [4] , where the number of relations differs significantly between objects with different class labels. A second finding is that unpublished papers have fewer references on average than published papers. Last, is that published papers have more pages than unpublished ones. This correlates with the finding that published papers are revised more frequently. Likely, as a paper is revised, additional text is added and the number of pages grows. It is also possible that the unpublished papers are fleshed out to longer reports to send to a journal and then are more likely to be accepted.

5.1 Predicting Publication

For this task, we trained two types of relational models, RPTs and relational multiple-instance learning [9] (RMIL), to predict whether a paper will be published in a journal. As explained in section 3.3, our analysis is limited to papers submitted to *arXiv* from 1995 to 2000. To classify a paper, the models used only information available at the time that the paper was submitted.

As a preliminary analysis, we attempted to differentiate between unpublished papers and papers published in *Physics Letters B*, the most common publication venue for

[2]This analysis applied the hubs and authorities algorithm to the undirected co-author graph. Hub and authority scores are equivalent on undirected graphs, and we choose to refer to the resulting scores as "authority scores".

arXiv papers. We sampled a set of 500 papers per year (3000 total), with equal proportion of published and unpublished papers. Given the difficulty of this task, the RPT performed well, with an average of 68% accuracy and 0.75 area under the ROC curve. The model selected four attributes that discriminate between unpublished and published papers: the number of authors, the number of references, the paper's length and the paper's filesize.

Figure 9a shows an example of a probability estimation tree learned by the algorithm. According to the model, published papers tend to have more authors and more references than unpublished papers, illustrated in Figure 8a and b.

The algorithm also distinguished between published and unpublished papers by size, measured in both kilobytes (KB) and number of pages. Figure 8d shows the distribution of paper length for published and unpublished papers in our sample of 3000 papers. The graph clearly shows that most *Physics Letters B* papers are between 5 and 15 pages in length, whereas the unpublished papers have widely varying lengths. The tree (shown in Figure 9b) predicts that papers over 16 pages in length and at least 13K in size were unlikely to be published (P(+)=0.03). After browsing a small subset of these papers on *arXiv*, it appears that the unpublished papers in the sample are either workshop papers (short papers, few references) or theses (long papers, a single author).

We also trained an RPT on the entire set of published and unpublished papers, and had moderately successful results (0.70 area under the ROC curve). The sample for each year had between 2300 and 3100 papers, and approximately 75% of the papers each year are published. The algorithm learned similar trees as the one learned for the previous task. As shown in Figure 8c, paper length is not as discriminative in this larger sample, which perhaps explain the lower performance on this larger set.

For RMIL, we created random samples of 200 papers (100 published and 100 unpublished papers) per year. RMIL achieved an accuracy of 61% with an average AUC of 0.61. RMIL identified that papers with 2 authors, papers that cited papers published in *Nuclear Physics B*, and papers that were cross-posted to areas other than *hep-th* were all more likely to be published.

5.2 Predicting Publication Venue

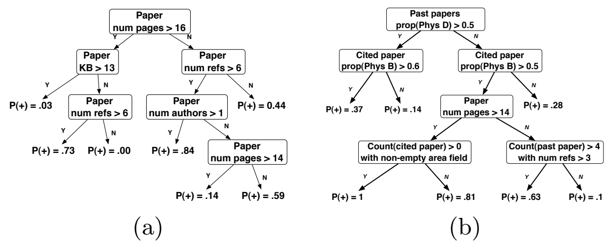


Figure 9: (a) RPT to predict whether a paper will be published in *Physics Letters B*. (b) RPT to predict between two popular journals.

We also trained an RPT for a related task, to differentiate between papers published in one of two popular journals (*Nuclear Physics B. Particle Physics, Field Theory and Statistical Systems, Physical Mathematics* and *Physical Review D. Particles, Fields, Gravitation, and Cosmology*). These are two of the most prevalent journals in the *arXiv* database. We expected this task to be challenging because approximately 55% of the papers were written by authors who have publications in both journals.

For each year, we sampled a set of 480 published papers, half of which were published in *Nuclear Physics B* and half in *Physical Review D*. For this task, RPTs achieved an average accuracy of 73% and an average AUC of 0.81 (see Table 8 for complete results). An example tree is shown in Figure 9b. The authors' publication history, the cited papers, and paper length are useful features to differentiate between papers published in these two journals. For example, if over 50% of an author's past papers were published in *Physics Letters D*, and less than 60% of cited papers were published in *Nuclear Physics B*, then the paper is unlikely to be published in *Nuclear Physics B* (P(+)=0.14).

6. COMMUNITY ANALYSIS

The final part of our analysis focused on identifying research communities by identifying groups of topics and the authors who publish in those topics. Our first approach to community detection was to use a conventional data clustering algorithm that considered only the paper's textual information for grouping papers into topics. However, research papers contain multiple sources of information for identifying topics; both textual content and citation structure can be used for clustering the documents. Our second approach used a clustering algorithm that combined citation structure and data information. Our third approach to clustering examined the topics formed naturally by considering the papers associated with each journal as distinct topics.

For the text-only clustering, we clustered according to a TFIDF based measure of document similarity. The clustering algorithm is based on an extension to the Lemur Toolkit[3]. We created six clusterings using both the full paper text and the abstracts and varying the similarity threshold. The resulting topics have higher intra-cluster citations than expected by chance (i.e. papers cite papers within the same topic more often than papers in other topics). However, the topic labels are not autocorrelated through journals or authors. Since we expect authors and journals to publish papers from a relatively small set of topics, we view this lack of autocorrelation as evidence of poor topic detection and focused on using the relational citation information to produce better clusters.

Research topics should be identifiable through groups of papers with similar terms and many intra-group citations. The web retrieval community has proposed a number of clustering algorithms that attempt to exploit both document contents and link structure to automatically group web documents into topics. One approach is to define a new similarity metric between documents that incorporates link structure and then use standard data-clustering algorithms (e.g. [16], [10]). Another approach is to weight the web graph with term similarities and use conventional graph clustering algorithms (e.g. [2]). We use the latter approach to cluster *hep-th* research papers.

We based our second approach on previous work by [15] on spectral partitioning algorithms using a normalized cut objective function. We use the citation graph to cluster papers,

[3]For more information, see http://ciir.cs.umass.edu

but modulate the strength of citation relationships by the semantic relationship indicated through content similarity. Our algorithm is quite similar to the approach used by [2] to identify topics in sets of retrieved web pages. However, they incorporate additional non-local link information into their similarity metric through summary co-citation information. We expect our algorithm to identify communities: groups of papers that have similar content and are also highly interconnected.

We clustered a sample of 833 papers from the *hep-th* database containing all papers in the years 1995-2000 with more than 50 non-self citations. Our intention was to sample a small set of authoritative papers that are likely to define topics. The algorithm used the portion of the citation graph that involved the 833 papers, weighted by the cosine similarity between paper abstracts.

Journals may be useful for detecting topics because it is common for journals to specialize and focus on research in specific sub-fields. To investigate this, we examined the 20,826 papers in *hep-th* that have journal information available and clustered those papers into distinct topics as determined by journal of publication. We eliminated clusters that were too small to represent meaningful topics by requiring that clusters contained at least 0.05% of the papers in the collection.

6.1 Community detection

The spectral clustering technique, which examines both content and citation information, produced 14 clusters varying in size from 2 to 285 papers. The number of papers in each cluster is shown in Figure 10A. Table 8 includes randomly selected titles from four examples clusters for subjective evaluation.

Our goal in this task was to identify communities of research. Authors write multiple papers on the same topic and are more likely to collaborate with other authors from the same community. Journals generally focus on a small number of topics and often specialize in particular topics. Because of this, we expect research communities should be identifiable through authors and journals, in addition to papers.

As a preliminary assessment of topics detection, we evaluated the correlation of clusters labels through authors and journals. Paper topic is correlated with journal (corr=0.58). Paper topics are autocorrelated through journals (corr=0.56) and through authors (corr=0.54). These correlations indicate that topics are associated with particular journals, that journals are associated with particular topics and that authors are associated with particular topics. This is evidence that the topics successfully identify communities of research. Figure 10b illustrates the autocorrelation of topic through authors graphically, plotting the number of distinct topics per author. These data are measured over all 478 authors associated with the sample of 833 papers.

Because topics are autocorrelated through authors, we can use the clusters to naturally partition the authors into communities as well. To cluster the authors in relation to the paper clusters, we assigned each author to the their most prevalent cluster based on authorship. Ties were broken randomly. Each cluster in Table 8 is labeled with the most authoritative author associated with the cluster. We associated journals with topics in the same way, assigning each topic to its most prevalent journal. The associated journals are listed along with authors in Table 8.

Cluster 2 : Sumit R.Das (251), *Physical Review D*
Absorption of Fixed scalars and the D-brane Approach to Black Holes; Universal Low-Energy Dynamics for Rotating Black Holes; Interactions involving D-branes; Black Hole Greybody Factors and D-Brane Spectroscopy
Cluster 7 : Gary T.Horowitz (588), *Physics Letters B*
On D-Branes and Black Holes in Four Dimensions; The Black Branes of M-theory; Counting States of Near-Extremal Black Holes; Internal Structure of Black Holes
Cluster 10 : Juan M. Maldacena (1924), *Journal of High Energy Physics*
Field theory models for tachyon and gauge field string dynamics; Super-Poincare Invariant Superstring Field Theory; Level Four Approximation to the Tachyon Potential in Superstring Field Theory; SO(32) Spinors of Type I and Other Solitons on Brane-Antibrane Pair
Cluster 13 : Ashoke Sen (4683), *Nuclear Physics B*
Dynamics of Anti-de Sitter Domain Walls; Gravitational Stability and Renormalization-Group Flow; String Theory on AdS_3; The Holographic Bound in Anti-de Sitter Space

Table 8: Example paper titles grouped together by spectral clustering. The authors shown are those with the highest number of non-self citations to papers in the cluster (with this number in parentheses).

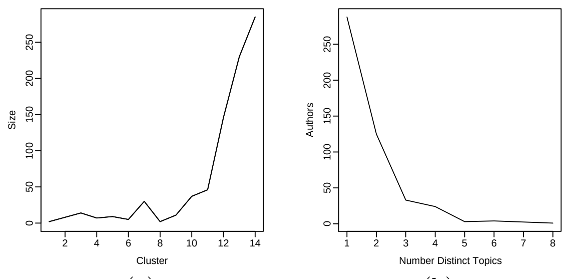


Figure 10: (a) number of papers per cluster, (b) Association of authors to paper clusters.

6.2 Topic analysis

We analyzed the topic clusters in several ways. We expect authors to cite papers within their own topic more than papers outside of the topic. For each of our topic clusterings, we calculated the actual and expected proportion of intra-cluster citations for each cluster. We define the actual proportion of intra-cluster citations for a cluster, C, as:

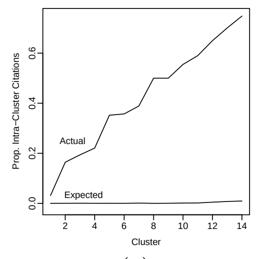
$$\frac{\text{\#of citations from C to C}}{\text{the total number of citations from cluster C}}.$$

We define the expected proportion of intra-cluster citations for a cluster, C, as:

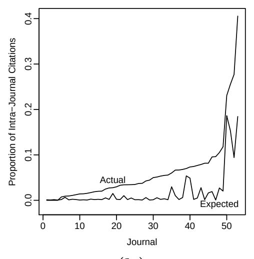
$$\frac{\text{the total number of papers in cluster C}}{\text{the total number of papers in the collection}}.$$

The expected proportion represents the proportion of intra-cluster citations we would expect given a uniform clustering across all topics.

Figure 11a shows the expected and actual intra-cluster citation proportions for the spectral clustering. For all but the smallest cluster, the proportion of intra-cluster citations is significantly higher than the expected values. This is not surprising, since the spectral algorithm is designed to minimize the normalized weighted-cut across clusters. We also calculated intra-journal citations in a similar manner. Figure 11b shows the expected and actual intra-journal citation proportions for each journal. As with the topic clusters, the actual intra-citation values deviate significantly from the expected values.The difference between the actual and the expected intra-clustering values demonstrates that the topics are cohesive with respect to citation patterns.

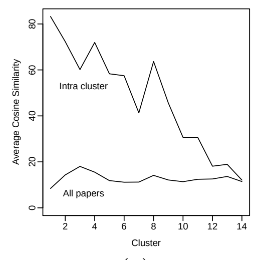


(a)

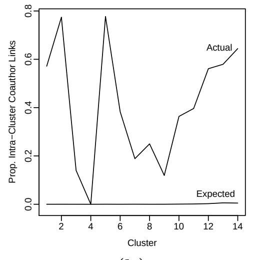


(b)

Figure 11: (a) Expected and actual intra-clustering citation ratios for spectral clustering (b) Expected and actual intra-journal citation ratios.



(a)



(b)

Figure 12: (a) Intra-cluster document similarity (b) Intra-cluster coauthor frequency.

To evaluate intra-textual similarity we averaged the cosine similarity across all pairs of documents within each cluster. As a baseline measure we averaged the cosine similarity between papers in a given cluster and all papers in the sample. Figure 12a plots the intra-cluster averages compared to the averages considering papers outside the cluster. For all but the largest cluster, the intra-cluster cosine similarity is much higher than expected, demonstrating that the topics are cohesive with respect to content. Average similarity may not the best measure to evaluate large clusters. Even when drawn from the same topic, it will be unlikely that all pairs of papers have similar content.

To evaluate whether the authors are more likely to collaborate within the clusters, we analyzed the coauthor links within clusters to see if the proportion of coauthor links within clusters was higher than expected. Figure 12b shows the expected vs. actual proportion of intra-cluster coauthor links. The zero value for cluster 4 is due to the fact that no authors were assigned to cluster 4 as their primary area. Collaboration is significantly higher with these clusters than would be expected by chance. This result further validates the claim that the spectral clustering has successfully identified research communities.

7. CONCLUSIONS

Based on our analysis, theoretical high-energy physics appears to be a healthy scientific community. Both the citation and authorship graphs reflect a pattern of tightly knit communication via the formal and informal scholarly literature. The community publishes a large numbers of papers, and the temporal pattern of citations indicates the rapid uptake and use of relevant new work. Despite the existence of “stars” such as Edward Witten, the papers of individual authors can vary greatly in their authority scores, indicating that papers are cited more for their innovative content than the pre-existing prominence of their author.

This analysis raises the possibility, already explored by the field of scientiometrics [14], of assessing and comparing the health of different scientific communities and subcommunities. The statistical techniques under development within relational learning offer an improved toolbox for the study of scientific networks, particularly as reflected in patterns of publication, citation, and downloading. Central to our analysis in this paper were: 1) techniques for calculating measures that use a combination of the attributes and structural of a relational data set; and 2) algorithms for learning statistical models that search a vast space of possible structures and parameter values to select those features most predictive of an attribute of interest. Both of these classes of methods allowed simultaneous consideration of multiple object and relation types, rather than only a single object and relation type, as is common in much prior work in citation analysis. Finally, consolidation of authors was important to the analysis above, and the relational structure was a strong contributor to how authors were consolidated.

Acknowledgments

We thank Hannah Blau for her comments and Daniel Larson for domain knowledge contributions. This effort is supported by DARPA, AFRL, and NSF under contract numbers F30602-00-2-0597, F30602-01-2-0566,and EIA9983215. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements either expressed or implied, of DARPA, AFRL, NSF or the U.S. Government.

8. REFERENCES

[1] H. Goldberg and T. Senator. Restructuring databases for knowledge discovery by consolidation and link formation. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*. AAAI Press, 1995.
[2] X. He, C. H. Q. Ding, H. Zha, and H. D. Simon. Automatic topic identification using webpage clustering. In *ICDM*, pages 195–202, 2001.
[3] D. Jensen and J. Neville. Linkage and autocorrelation cause feature selection bias in relational learning. In

Proc. of the 19th Intl Conference on Machine Learning, pages 259–266. Morgan Kaufmann, 2002.

[4] D. Jensen, J. Neville, and M. Hay. Avoiding bias when aggregating relational data with degree disparity. In *Proc. of the 20th Intl Joint Conf. on Machine Learning*, 2003.

[5] R. Kanigel. *Apprentice to Genius: The Making of a Scientific Dynasty.* Johns Hopkins University Press, 1993.

[6] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[7] P. Lazarsfeld and R. Merton. Friendship as social process: A substantive and methodological analysis. In *M. Berger et al. (eds.), Freedom and Control in Modern Society.* Octagon, New York", 1964.

[8] A. Lotka. The frequency distribution of scientific productivity. *Journal of the Washington Academy of Sciences*, 16:317–323, 1926.

[9] A. McGovern and D. Jensen. Identifying predictive structures in relational data using multiple instance learning. In *Proceedings of the Twentieth International Conference on Machine Learning*, 2003.

[10] D. S. Modha and W. S. Spangler. Clustering hypertext with applications to web searching. In *ACM Conference on Hypertext*, pages 143–152, 2000.

[11] J. Neville, D. Jensen, L. Friedland, and M. Hay. Learning relational probability trees. In *Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003.

[12] V. Pareto. *Le Cours d'Economie Politique.* Macmillan, London, 1897.

[13] L. Sachs. *Applied Statistics.* Springer-Verlag, 1982.

[14] Scientometrics: An international journal for all quantitative aspects of the science of science, communication in science and science policy. Kluwer Academic Publishers, Dordrecht, The Netherlands.

[15] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[16] R. Weiss, B. Velez, M. Sheldon, C. Namprempre, P. Szilagyi, A. Duda, and D. Gifford. Hypursuit: A hierarchical network search engine that exploits content-link hypertext clustering. In *ACM Conference on Hypertext*, Washington USA, 1996.

APPENDIX

A. CREATING THE SCHEMA

The data available for task 4 was in the form of LaTeX files, text abstract files, and the paper citations. From the abstract files, we extracted paper properties such as title, file size, journal reference, and submission dates. We used the earliest of the revision dates and the SLAC date as the best estimate of authorship date. Author names and institutions were parsed out of the Authors field, and the email address of the submitter was associated with the best-matching author name. Since institutions were not in a standardized format, we used the domain name of the submitter email address as a surrogate. Since the same authors, journals, and domains appear many times, we pulled them out into separate objects.

Journals were consolidated by hand; that is, we looked up their full names from the abbreviations, and coalesced differently-spelled references into the same object. The domains were given similarity links based on matching suffixes to facilitate identifying distinct institutions, and for use during author consolidation. We performed a nominal amount of hand data cleaning to correct for spelling errors or problems in formatting from the original submission form.

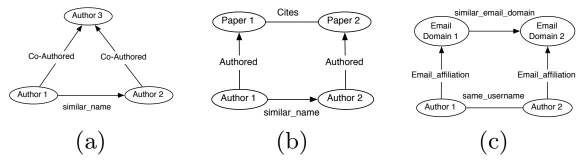


Figure 13: Relational evidence of duplicate authors. (a) Authors with a similar name who have co-authored with the same third-party. (b) Authors who have cited a paper written by an author with a similar name. (c) Authors with similar email domains and the same username.

B. AUTHOR CONSOLIDATION

Before analyzing the authors, we needed to identify duplicate author entries. Many *hep-th* authors publish under variants of the same name, e.g., "E.M.C. Abreu" and "Everton M.C. Abreu"; with other pairs like "J. Adams" and "J.A. Adams", the number of distinct identities was unclear. We began with the assumption that no two people had submitted papers under the same name (although this is rare, we did find a small number of instances in hand-checking the most frequent last names). We labelled pairs as *similar* if, after correcting for inconsistencies in punctuation and accents, the last names and the first initial of the first names matched. Of the initial 13,185 distinct author names, over 7500 had candidate matches to others.

Possible evidence for duplicate authors came from several sources. First, authors had to have *similar* names, and co-authors could never be consolidated. Another piece of evidence arose from author email addresses: using the same email address for multiple papers meant the authors were likely to be the same person. This was not conclusive evidence, because we found instances of people sharing email addresses. If a candidate pair's last name was rare (i.e. of the whole database, was only found on these two people), this boosted the evidence. For example, a number of people matched on the last name "Lee", but the only two authors with the last name "Znojil".

We also identified evidence for duplicate authors based on the relational neighborhood of the authors, as depicted in Figure 13. If two authors with similar names had each coauthored with the same third person, the two were likely to be the same person. Similarly, since people frequently cite their own work, we reasoned that if an author cites someone with a similar name, the two may well be the same person. Last, if two authors had the same username at similar email domains, this was considered to be comparable to using the same email address.

Using these guidelines, we iteratively identified and consolidated duplicate authors until quiescence. Because evidence involving third party authors was often not available until the third parties had themselves been merged correctly, this took five rounds of consolidation. At completion, we had 9200 distinct authors. Due to the noisy nature of the data, the final author set is not likely to be perfectly accurate, but as an example, it correctly merged all eight variations of the name "Ian Kogan," and of the top ten authors from Table 7, they were spread across 11 author objects (i.e. one mistake) instead of an initial 28. In addition, while the initial author graph contained 2206 connected components, after consolidation that number decreased to 1269.

Identifying Predictive Structures in Relational Data Using Multiple Instance Learning

Amy McGovern AMY@CS.UMASS.EDU
David Jensen JENSEN@CS.UMASS.EDU
Knowledge Discovery Laboratory, Computer Science Dept., Univ. of Massachusetts Amherst, Amherst, MA 01003, USA

Abstract

This paper introduces an approach for identifying predictive structures in relational data using the multiple-instance framework. By a predictive structure, we mean a structure that can explain a given labeling of the data and can predict labels of unseen data. Multiple-instance learning has previously only been applied to flat, or propositional, data and we present a modification to the framework that allows multiple-instance techniques to be used on relational data. We present experimental results using a relational modification of the diverse density method (Maron, 1998; Maron & Lozano-Pérez, 1998) and of a method based on the chi-squared statistic (McGovern & Jensen, 2003). We demonstrate that multiple-instance learning can be used to identify predictive structures on both a small illustrative data set and the Internet Movie Database. We compare the classification results to a *k*-nearest neighbor approach.

1. Introduction

Identifying useful structures in large relational databases is a difficult task. For example, consider the task of predicting which movies will be nominated for academy awards every year. The Internet Movie Database (IMDb) contains about one hundred movies that were nominated for academy awards in the time period 1970 to 2000 and thousands of movies that were not nominated in this time period. We would like to identify relational structure from a set of positive and negative examples (e.g., the structure surrounding nominated and non-nominated movies) that can explain known labels and predict labels for unseen data. Specifically, given the schema for the IMDb shown in Figure 1, we would like to identify some substructure that can predict which movies will be nominated and which movies will not be nominated. An example substructure could be a movie where one of the actors was previously nominated for an academy award. Such structures are useful not only for classification and prediction tasks but also for better understanding of large relational databases.

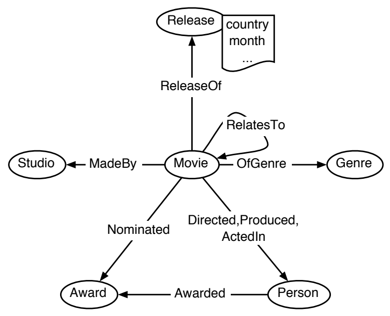


Figure 1. Schema that we used for the IMDb

Multiple instance learning (MIL) (details of MIL are given in Section 2) is a promising framework for identifying predictive structures in large relational databases. First, MIL methods are designed for learning from ambiguous and partially labeled data. With relational data, it is often easy to label a collection of objects and their relations. However, labeling each individual object and relation by its contribution to the overall situation is more difficult. For example, we can obtain the labels for the movie subgraphs by noting whether the movie was nominated for an academy award, but it would be difficult to label each actor and studio by their individual contribution to whether the movie was nominated for an award. Second, multiple-instance (MI) techniques are designed to identify which part of the data can explain the labels. For example, the relations in the movies example could contain all related movies, releases, studios, etc., for each nominated movie, but the best concept might only use the studio and producers linked via a movie.

MIL has been used successfully in a number of applications using propositional data (Amar et al., 2001; Dietterich et al., 1997; Goldman et al., 2002; Maron, 1998; Maron & Ratan, 1998; Zhang & Goldman, 2002; Zucker

& Chevaleyre, 2000). However, none of these techniques have examined MIL approaches for relational data even though the data set used in the introductory MIL paper (Dietterich et al., 1997) was relational (it was flattened into feature vectors to solve the task). By working with the data in relational form, we can detect structures that cannot be represented in a feature vector format. For example, a link between two related movies where the movie's producer was also nominated for an academy award for a previous movie would be very difficult to represent in propositional data, especially if the form of the final structure is not known in advance. Simply flattening the relational data (into homogeneous feature vectors) presents a number of problems. The homogeneity of the resulting data either means data duplication (which will affect probability estimation) or data loss through aggregation.

2. Notation and background

We use the PROXIMITY[1] system to represent, store, and query relational data sets. Let $G = (v, e)$ be a graph. Objects in the world, such as people, places, things, and events, are represented as vertices in the graph. Relations between these objects, such as *acted-in(actor, movie)* are represented by edges. In general, if there is a relation $r(o_1, o_2)$, then $o_1, o_2 \in v$ and $r \in e$. In PROXIMITY, vertices are called objects and relations are called links. Both objects and links can have multiple attributes associated with them. For example, using the schema shown in Figure 1, movies, people, genres, etc., are all objects. Relationships such as *awarded(movie, best-picture)* are links. Attributes can be associated with objects, such as *movie.name*, or links, such as *awarded.award-status*. PROXIMITY allows us to query the database using a graphical query language called qGraph (Blau et al., 2002). qGraph provides a form of abstraction on top of SQL by allowing us to construct visual queries of the graphical database. Queries return a collection of subgraphs and not just a set of database rows.

An MI learner uses labeled *bags* where a bag is a collection of *instances* with one label for the entire collection. A *positive bag* contains at least one instance of the *target concept* while a *negative bag* contains none. With flat data, both instances and target concepts are points in feature space. With relational data, both instances and target concepts are graphs, or relations among a (heterogeneous) set of objects. The goal is to find a concept that explains the labels for the bags and can be used to predict labels for unseen data. It is not known in advance which instance caused the bag to be labeled as positive. If this were known, a supervised learning approach could be used instead.

[1]For additional details on PROXIMITY, see http://kdl.cs.umass.edu.

We present an approach to adapting the MIL framework for use with relational data where bags are collections of graphs. The instances in the bag can be either explicitly enumerated as a set of graphs or they can be the set of (implicit) subgraphs of a single, larger, graph. The structure of the relational data determines which representation is most appropriate. If the data consist of sets of disjoint graphs, such as the MUSK task where each conformation of a molecule could be represented as a separate graph, then it is better to explicitly enumerate the instances in each bag. If the data consist of a large connected database, such as IMDb, then a bag consisting of single large graph can be easily created by querying the database. For example, in IMDb, the bags for the movies nominated for academy awards can be created by querying for all objects connected to a nominated movie by one or two links.

For the MI notation, we follow that of Maron (1998) and Maron and Lozano-Pérez (1998). The set of positive bags is denoted B^+ and the ith positive bag is B_i^+. Likewise, the set of negative bags is denoted B^- and the ith negative bag is B_i^-. If the discussion applies to both types of bags, we drop the superscript and refer to it as B. The jth instance of the ith bag is denoted B_{ij}. The target concept is denoted c_t and other concepts as c. With flat data, a concept is a point in feature space. With relational data, a concept is an attributed graph.

3. MI learning on relational data

The data available to an MI learner is a set of positive and negative bags, B^+ and B^-. If the concept is a feature vector v, then each bag consists of a set of feature vectors: $B_i = \{v_{i1}, v_{i2}, \ldots, v_{ik}\}$. The most straightforward transformation to apply MIL to relational data is to have each instance represented as a separate graph. In this case, a bag would consist of a set of graphs: $B_i = \{G_{i1}, G_{i2}, \ldots, G_{ik}\}$. The goal is then to find a concept that can explain the labeling of the bags. The concept, c, is a subgraph of one the graphs in B^+ and B^-. This representation is best suited for tasks where the data are already available as a set of disjoint graphs. The MUSK data set (Dietterich et al., 1997), image recognition tasks (Maron & Ratan, 1998), and the mutagenesis data set (Zucker & Chevaleyre, 2000) fit into this framework.

When the relational data are available as a large connected graph instead of a set of unconnected graphs, it may be easier to identify a single subgraph as containing something positive instead of enumerating every instance. For example, in the IMDb, we can hypothesize that there is some relational structure surrounding movies that could be used to predict whether a movie gets nominated for an academy award. Without knowing the structure in advance, it would be very difficult to create bags of every possible structure. However, it is relatively easy to identify the depth-two structure surrounding the movies and to use this to create bags where each bag has only one graph. The instances are assumed to be the set of all subgraphs of the single graph in the bag.

More formally, we propose to create the set of bags B^+ and B^- such that $B_i = \{G_i\}$ where G_i is a single (large) graph. The instances of B_i are assumed to be the set of all subgraphs of G_i. Since the size of this set is exponential in the size of G_i, where $|G_i|$ is defined as the sum of the number of vertices and edges in G_i, we do not explicitly enumerate the instances for each bag. Instead, the search methods take into account this assumption.

3.1. Relational diverse density

Several existing MI methods can be transformed to work with relational data. We adapt both diverse density (Maron, 1998; Maron & Lozano-Pérez, 1998) and chi-squared (McGovern & Jensen, 2003). We first briefly review the definitions for diverse density. The most diversely dense concept is defined as that which is closest to the intersection of the positive bags and farthest from the union of the negative bags. More precisely, Maron defines the diverse density of a particular concept c to be: $DD(c) = P(c = c_t|B^+, B^-)$. We refer to $P(c = c_t)$ as $P(c)$ to simplify the equations. Using Bayes rule and assuming independence, this can be reduced to finding the concept c for which the likelihood: $\prod_{1\le i\le n} P(c|B_i^+) \prod_{1\le i\le m} P(c|B_i^-)$ is maximal. The probability that concept c is the target concept given the evidence available in the bag, $P(c|B_i)$, still needs to be determined. Maron discusses several ways to do this. In this work, we follow his suggestion of using a noisy-or model (Pearl, 1988), in which case we have:

$$P(c|B_i^+) = 1 - \prod_{1\le j\le p} (1 - P(B_{ij}^+ \in c)) \qquad (1)$$

$$P(c|B_i^-) = \prod_{1\le j\le p} (1 - P(B_{ij}^- \in c)), \qquad (2)$$

where p is the number of instances in bag B_i and $P(B_{ij} \in c)$ is the probability that the specified instance is in the concept.

Calculating $P(B_{ij} \in c)$ requires a specific form of target concept. In the case of flat data, Maron often used what he called the single-point concept which is a point in feature space. With this concept, the calculation of $P(B_{ij} \in c)$ is based on the Euclidean distance between points B_{ij} and c in feature space. We need to define $P(B_{ij} \in c)$ when B_{ij} and c are both attributed graphs instead of points in feature space. To do this, we need a method for measuring the distance between two attributed graphs.

Metrics for measuring the distance between attributed graphs are not as well studied as metrics for flat data. We use the metric proposed by Bunke and Shearer (1998) which is based on finding the maximal common subgraph (MCS) between two graphs. They demonstrate that this distance measure satisfies the metric properties. The distance between two graphs G_1 and G_2 is defined as:

$$d(G_1, G_2) = 1 - \frac{|MCS(G_1, G_2)|}{\max(|G_1|, |G_2|)} \qquad (3)$$

where $MCS(G_1, G_2)$ is the maximum common subgraph of G_1 and G_2. This metric was developed for unlabeled graphs but can be modified so that the MCS also uses the attributes to limit the number of matches. A disadvantage of this metric is that computing the MCS is exponentially complete. In the course of a thorough search in concept space, MCS is calculated frequently. We approximate the calculation by limiting the depth of the recursive search. Research on a principled polynomial-time distance metric for attributed graphs is a topic for future work. Based on this metric, we define $P(B_{ij} \in c)$ as:

$$P(B_{ij} \in c) = \frac{|MCS(B_{ij}, c)|}{\min(|B_{ij}|, |c|)} \qquad (4)$$

Note that Equation 4 is a slight modification of Equation 3 where the maximum is replaced by a minimum. Since we are searching for the best subgraph, it is better to weight the match by the size of the proposed subgraph rather than by the size of the instances or of the bag, which could be arbitrarily large. If the instances in the bag are not enumerated, $P(c|B_i)$ becomes:

$$P(c|B_i^+) = \frac{|MCS(c, B_i^+)|}{\min(|c|, |B_i^+|)} \qquad (5)$$

$$P(c|B_i^-) = 1 - \frac{|MCS(c, B_i^-)|}{\min(|c|, |B_i^-|)}. \qquad (6)$$

This means that the probability that c is the correct concept given the evidence available in positive bag B_i^+ is the percent match of graph c to graph B_i^+. Likewise, the probability that c is the correct concept given the evidence in negative bag B_i^- is one minus the percent match of graph c to graph B_i^-. In other words, if c matches highly with B_i^+, the probability that c is correct will be high but if it matches highly with B_i^-, the probability that c is correct will be low.

3.2. Relational chi-squared method

In addition to diverse density, we present results using the chi-squared MI method (McGovern & Jensen, 2003). Chi-squared is simpler to calculate than diverse density and it allows for a more thorough search of the concept space because it provides a guaranteed pruning method. Chi-squared is calculated by filling in the contingency table shown in Table 1. The rows of the table correspond to the

Table 1. Contingency table used by the chi-squared method. The cells are filled in using the predicted and known labels for the training bags using the proposed concept.

		Actual Bag label	
		+	-
Predicted bag label	+	a	b
	-	c	d

predicted label from the concept and the columns correspond to the actual labels for the training bags. Assuming a method for labeling the bags given a proposed target concept, the table is filled out in the following manner. If the concept predicts that the bag will be positive and it is positive, *a* is incremented. If the prediction is positive but the bag is really negative, *b* is incremented. If the prediction is negative and the bag is positive, *c* is incremented. Finally, *d* is incremented if the concept predicts negative and the bag is negative. Chi-squared is calculated by summing the squared differences for the expected values in each cell of the contingency table versus the observed values.

The best concept is defined as that with the highest chi-squared value. This will occur when the mass is concentrated on the main diagonal (e.g., in *a* and *d*) which means that the concept is predicting the most positive and the most negative bags correctly. More information about the chi-squared evaluation function for MIL can be found in (McGovern & Jensen, 2003).

4. Experimental results: illustrative data set

We first present results using a small illustrative database where we both know the target answer in advance and can easily visualize the data. The objects and links each have one real-valued attribute associated with them. The target concept, shown in Figure 2, is a size-three clique with a particular set of attribute values on the objects and links.

We illustrate both chi-squared and diverse density using both data representations and this target concept. In both cases, graphs, including objects, links, direction of the links, and attributes, were generated randomly. To create a positive instance, a graph was randomly grown from the target clique. Negative instances were randomly grown from an empty graph. Attribute values from the target concept can be used in negative instances so long as the entire concept is not included. For the first data representation, both positive and negative graphs varied in size from three to ten objects with the same number of random links. Each positive bag had one positive instance and from two to six negative instances. Negative bags contained from three to seven negative instances. A sample positive instance and

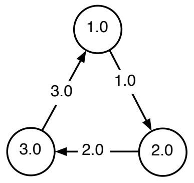


Figure 2. Target concept for the illustrative data set

a sample negative instance are shown in Figure 3a. The bags for the second data representation, which contained only one instance per bag, varied in size from ten to twenty objects and had twice as many random links as there were objects. Example positive and negative bags for this framework are shown in Figure 3b. In both cases, we generated twenty positive bags and twenty negative bags.

For this experiment, we compared the relative prediction accuracies for the relational diverse density approach, the chi-squared technique, and the *k*-nearest neighbors (kNN) method. We repeat this comparison for both data representations. For the diverse density and chi-squared approaches, the MI learner identified the best concept (or set of concepts) for predicting the bag labels. Given a relational concept c and a bag B_i with an unknown label, the predicted real-valued label is:

$$\text{label} = \max_{1\leq j\leq k} P(B_{ij} \in c) \quad = \quad \max_{1\leq j\leq k} \frac{|MCS(B_{ij}, c)|}{\min(|B_{ij}|, |c|)}.$$

If there is only one graph in the bag, this becomes:

$$\text{label} = \frac{|MCS(B_i, c)|}{\min(|B_i|, |c|)}.$$

Under this formulation, the predicted label for the bag will be a real number in the interval $[0,1]$. A prediction of zero means the bag should be labeled as negative and a prediction of one means that the bag should be labeled as positive. Values in the range $[0,1]$ are also possible and we examine the best choice of thresholds through the use of an ROC curve that measures the ratio of true positives to false positives as the threshold varies from zero to one.

We used kNN as a baseline for comparison. We identify the *k* nearest neighbors using the distance metric specified in Equation 3. Because the true labels for the individual instances are unknown, multiple instances in a bag are all assumed to have the same label as the bag. If the instances are not individually enumerated, we assign the label to the graph representing the bag itself and use this larger graph for the kNN calculations. We modify the prediction mechanism of kNN in the following manner. For each instance in an unlabeled bag, we determine the ratio of positive instances in the *k* nearest instances. The most extreme of these ratios weighted by the number of different positive or

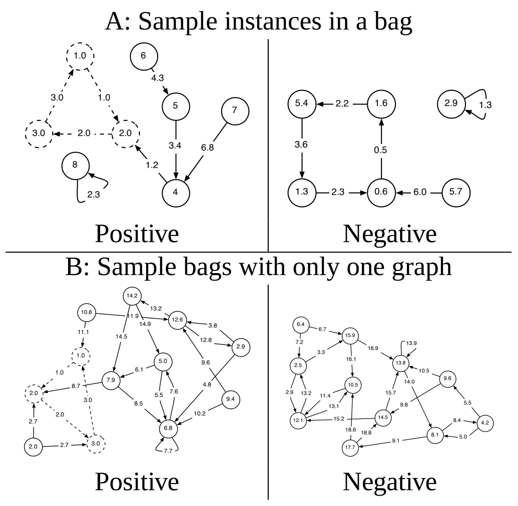

A: Sample instances in a bag
Positive
Negative
B: Sample bags with only one graph
Positive
Negative

Figure 3. A: Example instances for the three-clique task for the representation where each instance is enumerated. The target concept is shown with dashes. B: Example bags for the three-clique task for the representation where each bag consists of a single large graph.

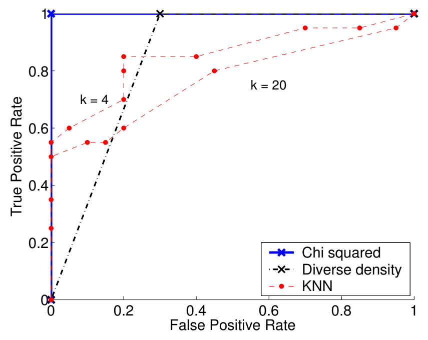


Figure 4. ROC curve comparing the performance of chi-squared, diverse density, and kNN on the illustrative data set. In this case, each bag had an enumerated set of instances.

negative bags that contributed to the ratio is chosen and the ratio itself (without the weighting) is output as the label. The idea of weighting the ratio this way is related to diverse density and helps to make kNN a higher performing baseline for comparison.

Figure 4 shows the ROC curves for relational diverse density, chi-squared, and two values of k for kNN for the first data representation, where there are multiple enumerated instances per bag. These numbers are averaged over 10-folds of cross validation. The test set for each fold was 2 positive bags and 2 negative bags and the training set was the remaining 18 positive bags and 18 negative bags. The chi-squared method identifies the correct target concept each time and had perfect prediction for this task. We do not claim that the chi-squared method will always have perfect performance but it was able to quickly find the target concept for this task. The relational diverse density approach sometimes found a subset of the true concept which gave it a small false positive rate depending on the threshold chosen to differentiate between positive and negative predictions. The two kNN approaches shown had higher accuracy than diverse density for very high thresholds but quickly degraded in performance while diverse density was more robust to threshold changes. At a threshold of 0.5, the accuracies were: chi-squared = 100%, diverse density = 85%, and kNN = 80% and 70% for $k = 4$ and $k = 20$. Accuracy is the percent of bag labels in the test set that are predicted correctly.

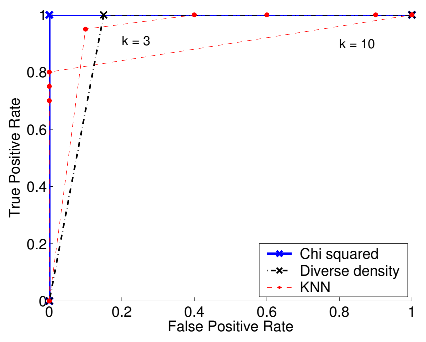
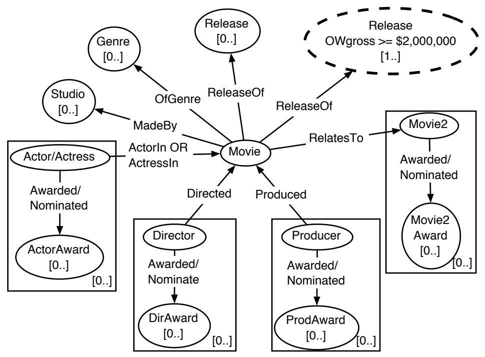


Figure 5. ROC curves for the illustrative data set where each bag had one large graph.

Figure 5 shows the ROC curve for the same three methods in the case where each bag had only one large graph in it. With a threshold of 0.5, chi-squared had 100% accuracy, diverse density had 92.5% accuracy, and kNN had 70% and 50% accuracies for $k = 3$ and $k = 10$. These results are comparable with those presented above and demonstrate that both data representations can be used successfully for MIL on relational data. Our next experiments focus on a much larger database.

5. Experimental results: IMDb

The IMDb is a much larger database with one million objects and nearly five million links. This is a large database where the ability to identify predictive structures should help us to better understand the nature of the database. The two tasks that we present are: predicting which movies will be nominated for academy awards and predicting which movies will gross at least two million dollars (adjusted for inflation) during opening weekend. Both of these tasks are very difficult and if there were a perfect predictor of movie success, then studio executives would have identified it already. Also, both tasks rely on an unknown number of factors which may not all be in the database (e.g., Hollywood politics are not included in IMDb). However, the difficulty



Query constraints:
movie2.year, release.year, Actor-award.year $<$ movie.year
Director-award.year, Producer-award.year $<$ movie.year
$1970 \leq$ movie.year ≤ 2000

Figure 6. qGraph query used to identify high-grossing movies and to create the positive bags. Dashed circles indicate the query restriction and number ranges indicate the minimum number of objects required for a match.

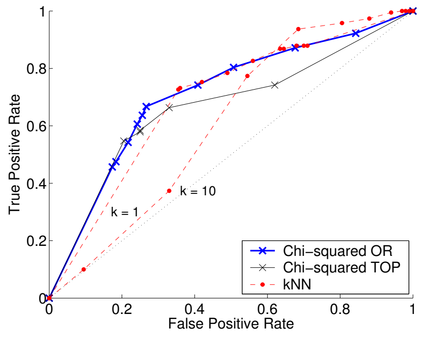


Figure 7. ROC curves comparing the false positive and true positive ratios for the chi-squared MI technique and kNN on the task of predicting high-grossing movies.

of the tasks provides a good challenge for our approach.

5.1. High-grossing movies

The IMDb is a large connected database and thus corresponds to the second data representation where each bag contains only one instance. We created the bags by querying the database using the qGraph query shown in Figure 6. This query is the depth-two structure surrounding high-grossing movies with the exception that we do not follow links from studios. Studios typically make hundreds of movies and following those links would lead to unnecessarily large graphs. This query returns a set of subgraphs from the database that match the specified structure. In particular, each subgraph will contain a central movie object and its related release objects where at least one release grossed more than 2 million dollars on opening weekend. In addition, any associated studios, genres, producers, directors, actors/actresses, and related movies will be included in each subgraph. If any of the producers, directors, actors/actresses, or related movies have award objects linked to them, these will also be included. Finally, the graph is pruned to remove any events that occurred after the movie's release. This is necessary because we want the structures that the MI learner identifies to predict forward in time. To help minimize noise and the size of the data, we further restrict the set to only contain movies from 1970 to 2000. We randomly sampled this set to obtain approximately 200 positive instances. We reused the same query structure to generate the negative bags except that the releases on opening weekend were restricted to gross less than 2 million dollars. There are a considerable number of such movies so we randomly subsampled to obtain approximately 200 negative bags.

We again ran 10-fold cross validation and obtained predictions for the unseen positive and negative bags from the top five percent of the concepts identified by MIL where the concepts are ranked by their chi-squared values. The inability to prune with diverse density hinders its use on such a large data set so we used only the chi-squared approach.

Figure 7 shows the results of this experiment using ROC curves. The chi-square method was able to detect several substructures that predicted high-grossing movies. The results shown in this graph are for the most highly ranked concept on each of the 10 folds, labeled chi-squared TOP, and for the top 5% of the concepts, labeled chi-squared OR. In the latter case, each concept outputs a separate prediction and we used the OR, or max, of these predictions. Although MIL has slightly lower performance in the region of the ROC curve with higher true positives but also higher false positives, its performance is better than kNN in the region with lower false positives and higher true positives. Also, its performance only degrades as the threshold is dropped almost to zero while kNN is less robust to the threshold value. With a threshold of 0.5, chi-squared TOP achieves an accuracy of 69.2% and chi-squared OR has a 70.1% accuracy. kNN's accuracies are 61% and 53.6% for $k = 1$ and $k = 10$. With this prediction mechanism, studios could better allocate money to movies. As we said in the beginning of this experiment, predicting high-grossing movies is a difficult task and it is unlikely that any learning agent could achieve high accuracy.

One of the other benefits of using MIL on this database, besides prediction accuracy, is that the answers are in the form of understandable structures. Figure 8 shows some of the top structures for predicting high-grossing movies. It seems that movies are more likely to be high-grossing if they are related to two or three other movies (e.g. a movie

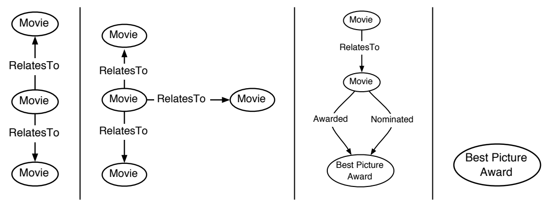


Figure 8. Top predictive relational structures identified by the MI learner on the high-grossing movie task.

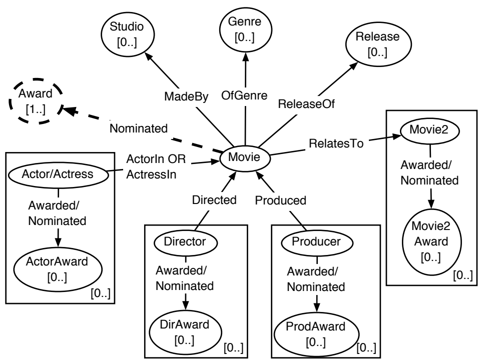
in a series like *Star Trek* or *Indiana Jones* or movies that remade previous successful movies). Another predictive structure that the chi-squared MI learner identified was a movie related to another movie that was both nominated and awarded an academy award for best picture. Last, just the presence of a best picture award object in the subgraph was predictive of movie success.

5.2. Movies nominated for academy awards

We repeated the same experiments for the difficult task of predicting which movies will be nominated for academy awards each year. The query used to generate the positive bags is shown in Figure 9. The structure of this query is identical to that discussed for high-grossing movies except that we require an academy award nomination. The positive bags do not actually contain the award objects for the central movie because we want MIL to identify predictive structures. This query yields 72 positive bags. We use the same query minus the requirement for the awards to create the negative bags. The number of movies which are not nominated for academy awards is quite large and we randomly sample this set to obtain approximately the same number of negative bags (74).

We again compare the predictive ability of chi-squared to kNN on 10-fold cross validation with this data set. These results are shown in Figure 10, again using ROC curves. In this case, the structures found by the MI learner dominate any of the predictions from kNN for all values of *k* (We show two of the best values of *k* in the figure). Assuming a threshold of 0.5, the accuracy of chi-squared TOP is 93% and the accuracy of chi-squared OR is 77%. kNN has an accuracy of 49.7% and 50.7% for $k = 5$ and $k = 10$.

We also examine the relational structures that the MI learner identified as predictive of whether a movie will be nominated for an academy award. Some of the top structures are shown in Figure 11. For this task, it seems that movies with at least 20 actors in them are more likely to be nominated for academy awards. This is surprising and is likely due to a reverse effect that better movies have more information in IMDb which means they tend to have more actors associated with them. A related structure has the same form but restricts the genre to drama. These structures



Query constraints:
movie2.year, release-year, Actor-award.year $<$ movie.year
Director-award.year, Producer-award.year $<$ movie.year
$1970 \leq$ movie.year ≤ 2000

Figure 9. qGraph query used to create the positive bags for the task of predicting which movies will be nominated for an academy award.

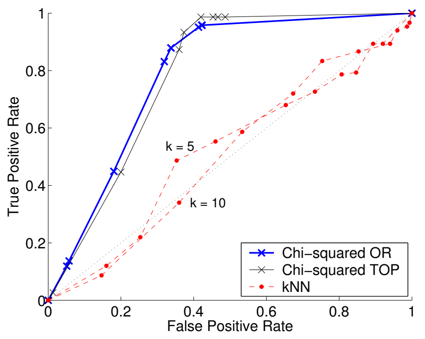


Figure 10. ROC curves for the task of predicting which movies will be nominated for academy awards.

may not help a studio executive to better allocate money to new movies but they did identify an important characteristic of the database, which is one of the goals of this work. The presence of only a drama object is enough to predict the nomination in many cases. Last, if a previous award object existed in the subgraph, e.g., if the movie was related to a movie that was also nominated or won an academy award, it was likely to be nominated itself.

6. Discussion and Conclusions

In this paper, we have presented an approach to identifying predictive structures in relational databases based on the MIL framework. We adapted this framework for use with relational data in two related ways: one where the bags had multiple independent graphs as the instances and one where the bags had one larger graph and the instances were the (implicit) subgraphs of this graph. We demon-

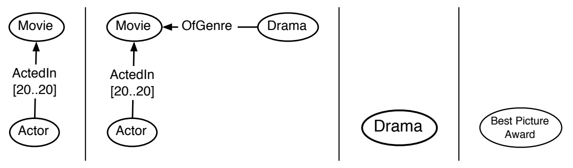


Figure 11. Relational structures identified by the MI learner for predicting academy award nominees.

stated that these adaptations could be used to modify existing MI methods and that the relational version of these methods could be used successfully on both a small and a large database.

One of the strengths of MIL that is emphasized for flat data is the ability to identify which features of the task are important. In the diverse density framework, this is referred to as *scaling*. When the concept is a feature vector, diverse density can identify a scale for each feature that maximizes the diverse density value. If a feature is irrelevant, its best scale will be zero. This strength also applies to the techniques that we presented in this paper. Instead of scaling features in a vector, the concepts identified by the relational MI learner will only contain a subset of the objects and links from the bags. This subset represents the more relevant features with respect to the current task.

Another advantage of MI techniques is that they identify an actual concept (or set of concepts) that can be understood by a human. kNN can be used to label new data but it cannot identify aspects of the data that can help a human to better understand the database. With such structures, a human can iteratively refine their understanding of the database and of the tasks at hand.

Relational probability trees (RPT) (Neville et al., 2003) are a related approach in that they have also been developed to identify predictive structure in large relational databases. However, MIL and RPTs express different relational concepts. RPTs are designed to identify structure in a tree form using attributes on objects or links or structure such as the number of outgoing links from an object. Although this can work very well on tasks such as predicting high-grossing movies, RPTs cannot represent graph concepts such as the 3-clique presented in Section 4.

Acknowledgments

The authors are grateful for comments and helpful discussions from Kiri Wagstaff, Jennifer Neville, Hannah Blau, and the anonymous reviewers. This effort is supported by DARPA and AFRL under contracts F30602-00-2-0597 and F30602-01-2-0566. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements either expressed or implied, of DARPA, AFRL, or the U.S. Government.

References

Amar, R. A., Dooly, D. R., Goldman, S. A., & Zhang, Q. (2001). Multiple-instance learning of real-valued data. *Proc. of the 18th International Conference on Machine Learning* (pp. 3–10). Morgan Kaufmann, SF, CA.

Blau, H., Immerman, N., & Jensen, D. (2002). *A visual language for querying and updating graphs* (Technical Report 2002-037). University of Massachusetts Amherst.

Bunke, H., & Shearer, K. (1998). A graph distance metric based on the maximal common subgraph. *Pattern Recognition Letters*, *19*, 255–259.

Dietterich, T. G., Lathrop, R. H., & Lozano-Perez, T. (1997). Solving the multiple-instance problem with axis-parallel rectangles. *Artificial Intelligence*, *89*, 31–71.

Goldman, S., Zhang, Q., Yu, W., & Fritts, J. E. (2002). Content-based image retrieval using multiple-instance learning. *Proc. of the 19th International Conference on Machine Learning* (pp. 682–689). Morgan Kaufmann, SF, CA.

Maron, O. (1998). *Learning from ambiguity*. Doctoral dissertation, Massachusetts Institute of Technology.

Maron, O., & Lozano-Pérez, T. (1998). A framework for multiple-instance learning. *Advances in Neural Information Processing Systems 10* (pp. 570–576). Cambridge, Massachusetts: MIT Press.

Maron, O., & Ratan, A. L. (1998). Multiple-instance learning for natural scene classification. *Proc. of the 15th International Conference on Machine Learning* (pp. 341–349). Morgan Kaufmann, San Francisco, CA.

McGovern, A., & Jensen, D. (2003). *Chi-squared: A simpler evaluation function for multiple-instance learning* (Technical Report 03-14). Computer Science Department, University of Massachusetts Amherst.

Neville, J., Jensen, D., Friedland, L., & Hay, M. (2003). Learning relational probability trees. To appear in the *Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. San Mateo, California: Morgan Kaufmann Publishers.

Zhang, Q., & Goldman, S. A. (2002). EM-DD: An improved multiple-instance learning technique. *Advances in Neural Information Processing Systems 14*. Cambridge, MA: MIT Press.

Zucker, J.-D., & Chevaleyre, Y. (2000). *Solving multiple-instance and multiple-part learning problems with decision trees and decision rules. application to the mutagenesis problem* (Technical Report 6). University of Paris.

Collective Classification with Relational Dependency Networks

Jennifer Neville and David Jensen

Department of Computer Science
140 Governors Drive
University of Massachusetts, Amherst
Amherst, MA 01003
{jneville|jensen}@cs.umass.edu

Collective classification models exploit the dependencies in a network of objects to improve predictions. For example, in a network of web pages, the topic of a page may depend on the topics of hyperlinked pages. A relational model capable of expressing and reasoning with such dependencies should achieve superior performance to relational models that ignore such dependencies. In this paper, we present relational dependency networks (RDNs), extending recent work in dependency networks to a relational setting. RDNs are a collective classification model that offers simple parameter estimation and efficient structure learning. On two real-world data sets, we compare RDNs to ordinary classification with relational probability trees and show that collective classification improves performance.

1 Introduction

In this paper, we show how *autocorrelation* can be used to improve the accuracy of statistical models of relational data. Autocorrelation is a statistical dependency between the values of the same variable on related entities and is a common characteristic of many relational data sets. In previous work, we have shown how autocorrelation can cause bias in learning algorithms (Jensen & Neville 2002), particularly when it appears with *concentrated linkage*, another common characteristic of relational data sets. However, this work explores methods which exploit autocorrelation to improve model accuracy. In particular, we demonstrate this can be accomplished with a relatively simple approach to structure learning in a class of undirected graphical models that we call relational dependency networks (RDNs).

It is relatively easy to understand how autocorrelation could be used to improve the predictions of statistical models. For example, consider the problem of automatically predicting the *topic* of a technical paper (e.g., neural networks, reinforcement learning, genetic algorithms). One simple method for predicting paper topics would look at the position of a paper within a citation graph. It may be possible to predict a given paper's topic with high accuracy based on the topics of neighboring papers in this graph. This is possible only because we expect high autocorrelation in the citation graph—papers tend to cite other papers with the same topic.

However, such a scheme for prediction assumes that all the labels of related entities (e.g., topics of referenced papers) are known. In many cases, the topics of an entire set of papers may need to be inferred simultaneously. This approach, called *collective classification* (Taskar, Abbeel & Koller 2002) requires both models and inference procedures that can use inferences about one entity in a relational data set to influence inferences about related entities. Similar approaches have been used in several recent applications (Neville & Jensen 2000; Chakrabarti, Dom & Indyk 1998).

In this paper, we introduce relational dependency networks (RDNs), an undirected graphical model for relational data. We show how RDNs can be learned and how RDNs and Gibbs sampling can be used for collective classification. Because they are undirected graphical models, RDNs can represent the cyclic dependencies required to express autocorrelation, and they can express a joint probability distribution, rather than only a single conditional distribution. In addition, they are relatively simple to learn and easy to understand.

We show preliminary results indicating that collective inference with RDNs offers improved performance over non-collective inference that we term "individual inference." We also show that RDNs applied collectively can perform near the theoretical ceiling achieved if all labels of neighbors are known with perfect accuracy. These results are very promising, indicating the potential utility of additional exploration of collective inference with RDNs.

2 Classification Models of Relational Data

Many relational models are used for "individual inference," where inferences about one instance are not used to change the inference of related instances. For example, some work in inductive logic programming (ILP) and relational learning uses relational instances that can be represented as disconnected subgraphs (e.g., molecules), thus removing the need (and the opportunity) for collective inference. Such models can cope with complex relational structure *of an instance*, but they do not attempt to model the relational structure *among instances*.

Even learning relational models for individual inference can be difficult because learning in relational data differs substantially from learning in propositional settings. For example, work in ILP has long considered difficult representational issues such as recursion, aggregation, variables, quantification, and other aspects of first-order and higher-order logics. In addition, some of our recent work has examined the unique statistical properties of relational data, and the influence of these characteristics on learning algorithms. We have shown how concentrated linkage and relational autocorrelation can bias relational learners toward particular features (Jensen & Neville 2002). We have also shown how degree disparity can cause misleading correlations in relational data, leading learning algorithms to add excessive structure to learned models (Jensen, Neville & Hay 2003).

To address the challenges of relational learning, new learning algorithms are necessary. For example, probabilistic relational models (PRMs) (Getoor, Friedman, Koller & Pfeffer 2001) are a form of directed graphical model thatextend Bayesian networks to support reasoning in complex relational domains. Unfortunately, PRMs are a directed model in which autocorrelation cannot be represented due to acyclicity constraints.[1]

As another example, we have recently developed relational probability trees (RPTs), a method for learning tree-structured models that encode conditional probability distributions for a class label that depend on both the attributes of related instances and the features of the surrounding structure (Neville, Jensen, Friedland & Hay 2003). Our methods for learning RPTs adjust for the sources of bias mentioned above. As a result, the learned RPTs are a relatively compact and parsimonious representation of conditional probability distributions in relational data. Until recently, we had only applied these models for individual inference. The algorithm for learning RPTs adjusts for statistical biases caused by autocorrelation, but our inference techniques have not made use of autocorrelation to improve inference.

This is unfortunate, because autocorrelation is a nearly ubiquitous phenomenon in relational data. We have observed relatively high levels of autocorrelation in relational data sets. For example, in analysis of the 2001 KDD Cup data we found that the proteins located in the same place in a cell (e.g., mitochondria or cell wall) had highly autocorrelated functions (e.g., transcription or cell growth). Autocorrelation has been identified by other investigators as well. For example, fraud in mobile phone networks has been found to be highly autocorrelated (Cortes, Pregibon & Volinsky 2001). The topics of authoritative web pages are highly autocorrelated when linked through directory pages that serve as “hubs” (Kleinberg 2001).

We also know that exploiting autocorrelation can result in significant increases in predictive accuracy. Several recent developments in relational learning have focused on exploiting autocorrelation. For example, the relational vector-space (RVS) model (Bernstein, Clearwater & Provost 2003) uses weighted adjacency vectors to construct classifiers. The model is extremely simple, but it produces accurate classifiers in data with strong autocorrelation. Other examples include work with web pages (Chakrabarti et al. 1998) that uses the hyperlink structure to produce smoothed estimates of class labels and our own prior work (Neville & Jensen 2000) that uses an iterative classification scheme to improve accuracy by exploiting the inferred class labels of related instances.

Recently, undirected graphical models capable of representing and reasoning with autocorrelation have been explored for the task of modeling relational data. Domingos and Richardson (2001) represent market entities as social networks and develop Markov random field models to model the influence in purchasing patterns throughout the network. Taskar et al. (2002) use relational Markov networks (RMNs), based on conditional random fields for sequence data (Lafferty, McCallum & Pereira 2001), to model the dependencies among web pages to predict page type. Undirected models have proved to be successful for collective classification of relational data. However, research into these models has focused primarily on parameter estimation and inference procedures. Model structure is not learned automatically, it is pre-specified by the user. Also, the models do not automatically identify which features are most relevant to the task. For relational tasks, which are likely to have a large

[1] An exception is where autocorrelation is structured by some additional information such as temporal constraints. (Getoor et al. 2001)

number of features, this lack of selectivity will make the model more difficult to interpret and understand.

3 Relational Dependency Networks

Relational dependency networks extend recent work on dependency networks (Heckerman, Chickering, Meek, Rounthwaite, and Kadie 2000) to a relational setting. Dependency networks (DNs) are graphical models of probabilistic relationships—an alternative to Bayesian networks and Markov random fields. DNs are easy to learn and have been shown to perform well for a number of propositional tasks so we expect them to offer similar advantages when used in a relational setting. We begin by reviewing the details of dependency networks for propositional data and then describe how to extend dependency networks for use with relational data.

3.1 Dependency Networks

Like Bayesian networks and Markov random fields, dependency networks encode probabilistic relationships among a set of variables. Dependency networks are an alternative form of graphical model that approximate the full joint distribution with a set of conditional distributions that are learned independently. DNs combine characteristics of both undirected and directed graphical models. The dependencies among variables are represented with an undirected graph, so conditional independence can be interpreted using graph separation. However, as with directed models, dependencies are quantified using conditional probability distributions (CPDs) of a variable given its parents. The primary distinction between DNs and other graphical models is that DNs are an *approximate* model. Because the CPDs are learned independently, DN models are not guaranteed to specify a coherent probability distribution (see *Learning* section below for details).

DNs offer several advantages over conventional Bayesian networks, but also have several disadvantages (Heckerman et al. 2000). Unlike Bayesian networks, DNs are difficult to construct using a knowledge-based approach and they cannot represent causal relationships. However, DN models can encode predictive relationships (i.e. dependence and independence) and there are simple techniques for parameter estimation and structure learning of DNs.

The characteristics that distinguish DN models from Bayesian networks make them similar to Markov random fields. Both DNs and Markov random fields use undirected graphs to represent dependencies among variables. When the causal relationships among variables are unknown, undirected models are often more interpretable than directed models which require d-separation reasoning to assess conditional independencies. Undirected graphical models also have straightforward techniques for parameter estimation and structure learning (e.g. Della Pietra, Della Pietra and Lafferty 1997). DN conditional probability distributions will generally be easier to inspect and understand than Markov network clique potentials, but DNs approximate the full joint distribution and therefore Markov networks may produce more accurate probabilities.

Representation. The DN model consists of a set of conditional probability distributions (CPDs), one for each variable given its parents. Consider the set of variables $X=(X_1,...,X_n)$ with a joint distribution $p(x)=p(x_1,...,x_n)$. A dependency network for X is represented by a graph G where each node in G corresponds to an X_i□X. The parents of node X_i, denoted pa_i, consist of the nodes in its *Markov blanket*: This specifies that, given its parents, a node is conditionally independent of the rest of the nodes in the graph:

$$p(x_i \mid pa_i) = p(x_i \mid \mathbf{x} \setminus x_i)$$

The undirected edges of the graph connect each node x_i to each of its parents (the nodes in pa_i). Furthermore, each node in the graph contains a local CPD, $p_i=p(x_i|pa_i)$. Together these CPDs specify the joint distribution over $\boldsymbol{X}$.

For example, the DN in figure 1 could be used to model the set of variables $\boldsymbol{X}=(X_1,X_2,X_3,X_4,X_5)$. Each node is conditionally independent of the other nodes in the graph given its parents. For example, X_1 is conditionally independent of X_2 and X_4 given X_3 and X_5. Each node contains a CPD, which specifies a probability distribution over possible values given the values of its parents. The full joint probability is the product of the local CPDs:

$$p(X) = p(X_1 \mid X_3,X_5)p(X_2 \mid X_3,X_4)p(X_3 \mid X_1,X_2)p(X_4 \mid X_2,X_3)p(X_5 \mid X_1)$$

Notice that the model may have cycles. For example, X_2, X_3 and X_4 form a clique in the graph. This necessitates the use of approximate inference techniques (see *Inference* section below for details). Also, notice that X_4 is dependent on X_3, but the reverse is not true. An undirected edge is placed between two nodes if either of the nodes is dependent on the other. In this case, a node's parents will still form a Markov blanket, but they won't be the minimal set (e.g. $p_3=p(x_3|x_1,x_2,x_4)= p(x_3|x_1 ,x_2)$).

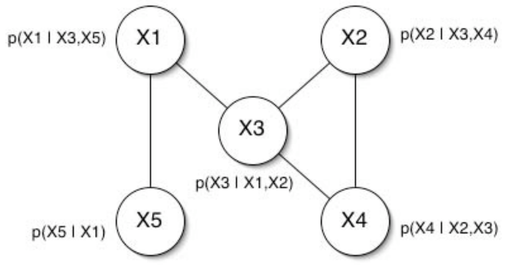


Fig. 1. Sample dependency network.

Learning. As with any graphical model, there are two components to learning a DN: structure learning and parameter estimation. Both structure learning and parameter estimation is accomplished through learning the local CPDs of each variable. The structure of a DN model is defined by the components of the local CPDs, much as feature specifications define the structure of a undirected graphical model. The edges of the graph connect a node to each of the variables in its local CPD. The parameters of the model correspond to the parameters of the local CPDs.

The DN learning algorithm learns a separate CPD for each variable, conditioned on the other variables in the data. For the DN model to be less than fully connected, a selective learning algorithm should be used. Any selective modeling technique can be used, however, to build a parsimonious DN model it is desirable to use a selective learner that will learn small, accurate CPDs.

For example, an algorithm for learning relational probability trees could be used to model X_1 given X_2,X_3,X_4,X_5. The variables included in the tree would then be designated as X_1's parents in the network, the appropriate edges would be added to the graph (e.g. X_1 to X_3 and X_1 to X_5) and the tree itself would be used as the local CPD for X_1. Learning the full DN in figure 1 would require learning five models, one tree for each variable in the graph.

Although the DN approach to structure learning is simple, it can result in an inconsistent network, both structurally and numerically—there may be no joint distribution from which each of the CPDs can be obtained using the rules of probability. Learning each local CPD independently may result in an inconsistent network where there is an edge between two variables but one is not dependent on the other (e.g. X_4 is dependent on X_3 but not vice versa). Independent parameter estimation may also result in inconsistencies where the overall joint distribution does not sum to 1. However, Heckerman et al. (2000) show that DNs will be "nearly" consistent if learned from large data sets. That is, the data serve a coordinating function that ensures some degree of consistency among the CPDs.

Inference. Unlike Bayesian networks, the dependency network graphs are potentially cyclic. This is due to the nature of the structure-learning algorithm where there is no restriction on the form of the CPDs. Cyclic dependencies necessitate the use of approximate inference techniques such as Gibbs sampling (e.g. Neal 1993) or loopy belief propagation (e.g. Murphy, Weiss and Jordan 1999). Heckerman et al. use Gibbs sampling to combine the models to approximate the full joint distribution and extract probabilities of interest. In practice, DNs have produced good approximations to the joint distributions represented by directed graphical models (Heckerman et al. 2000).

3.2 Relational Dependency Networks

Relational dependency networks (RDNs) extend DNs to work with relational data. The extension is similar to the way in which probabilistic relational models (Getoor et al. 2001) extend Bayesian networks for relational data.

Representation. RDNs specify a probability model over a network of instances. Given a set of objects and the links between them, a RDN defines a full joint probability distribution over the attribute values of the objects. Attributes of an object can depend probabilistically on other attributes of the object, as well as on attributes of objects in its relational neighborhood.

Instead of defining the dependency structure over attributes, as in DNs, RDNs define a generic dependency structure at the level of object *types*. Each attribute A_i associated with object type X is linked to a set of parents that influence the value of

A_i. Parents of A_i are either (1) other attributes associated with type X, or (2) attributes associated with objects of type Y where objects Y are linked to objects X. For the latter type of dependency, if the relation between X and Y is one-to-many, the “parent” consists of a set of attribute values. In this situation, RDNs uses aggregated features to map sets of values into single values.

For example, Figure 2 contains an example dataset from the movie domain. There are three types of objects—movies, studios and actors. Each object type has a number of associated attributes that will be each be modeled in the RDN. Consider the attribute *actor.hasAward*—the set of potential parents for this attribute consists of *actor.gender*, *movie.receipts* and *studio.country*. Notice that an actor can star in multiple movies and thus be associated indirectly with multiple studios.

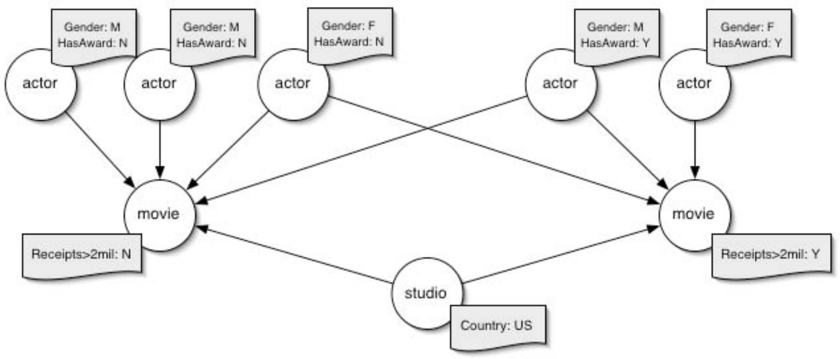


Fig. 2. Sample relational data graph.

The full RDN model is potentially much larger than the original data graph. To model the full joint distribution there must be a separate node (and CPD) for each attribute value in the data graph. Consequently, the total number of nodes in the model graph will be $\sum_{T} N_T A_T$ where N_T is the number of objects of type T in the data graph and A_T is the number of attributes for objects of that type. To make the models tractable, the structure and parameters of the CPDs are tied—the RDN model contains a single CPD template for each attribute of each type of object. For the example above, the model would consist of four CPDs, one for *actor.gender*, *actor.hasAward*, *movie.receipts* and *studio.country*.

To construct the model graph, the set of template CPDs is *rolled-out* over the entire data graph. Each object-attribute pair gets a separate, local copy of the appropriate CPD. This facilitates generalization across data graphs of varying size. We can learn the CPD templates from one data graph and apply the model to a second data graph with a different number of objects by rolling-out more CPD copies. This approach is analogous to other graphical models that tie distributions across the network (e.g. hidden Markov models, PRMs).

Figure 3 contains a possible RDN model graph for the example discussed above. It shows dependencies between *actor.gender* and *actor.hasAward*, between *actor.hasAward* and *movie.receipts*, and between *movie.receipts* and *studio.country*. Furthermore, there is a dependency between *movie.receipts* of related movies. Notice the hyper-edges between movies and associated actor awards. This indicates that movie receipts is dependent on an aggregated feature of *actor.hasAward* (e.g. EXISTS(*actor.hasAward*=Y)). Aggregation is one approach to ensure the template CPDs are applicable across data graphs of varying structure. Each movie may have a different number of actor award values, so aggregation is used to map these sets of values into single values.

Learning. Learning an RDN model again consists of two tasks: learning the dependency structure, and estimating the parameters of the conditional probability distributions. The RDN learning algorithm is much like the DN learning algorithm, except we use a selective *relational* classification algorithm to learn a set of conditional models. We use relational probability trees (RPTs) for the CPD components of the RDN. RPTs extend standard probability estimation trees to a relational setting in which data instances are heterogeneous and interdependent (Neville et al. 2003). RPT models estimate probability distributions over possible class label values in the same manner as conventional classification trees, but the algorithm looks beyond the attributes of the item for which the class label is defined and considers the effect of the local relational neighborhood on the probability distribution. RPT models represent a series of questions to ask about an item and the objects/links in its relational neighborhood.

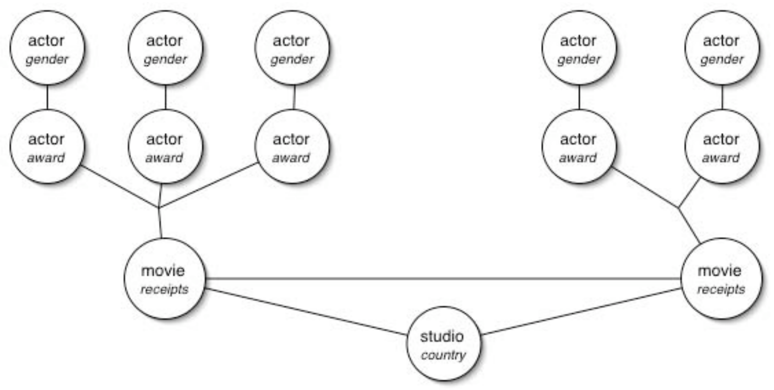


Fig. 3. Sample RDN model graph.

The RPT learning algorithm (Neville et al. 2003) uses a novel form of randomization tests to adjust for biases towards particular features due to the characteristics of the relational data (e.g. degree disparity, autocorrelation). We have shown that RPTs learned with randomization tests, build significantly smaller trees than other models and achieve equivalent, or better, performance. This characteristic of the RPTs is crucial for learning parsimonious RDN models—the collection of RPT models will be used during Gibbs sampling so the size of the models will have a direct impact on inference efficiency.

Given a data graph, an object type and a target attribute—an RPT is learned to predict the attribute given (1) the other attributes of the object, and (2) the attributes of other objects and links in the relational neighborhood. In our current approach, the user specifies the size of relational neighborhood to be considered by the RPT algorithm in (2). For efficiency reasons, we've limited the algorithm to consider attributes of objects one or two links away in the data graph. However, it is possible for the RPT models to consider attributes of much more "distant" objects (in the sense of a graph neighborhood).

Figure 4 shows an example RPT learned on data from the IMDb to predict whether a movie's opening-weekend receipts are more than $2million, given the attributes of movies and everything up to two links away—actors, directors, producers and studios, as well as movies associated with those objects (see Section 4 for experimental details). The tree indicates that movie *receipts* depend on the receipts of other movies made by the same studio, as well as actor age and movie genre. The root node of this tree asks whether more than 60% of the other movies made by the studio (e.g. *Studio Movie* objects) have *receipts=Y*. If this is not the case, the model moves to the next node on the right branch and asks whether the movie has more than 5 actors born after 1958. If the answer to this question is also no, a prediction of *P(Y)*=0.01 is returned.

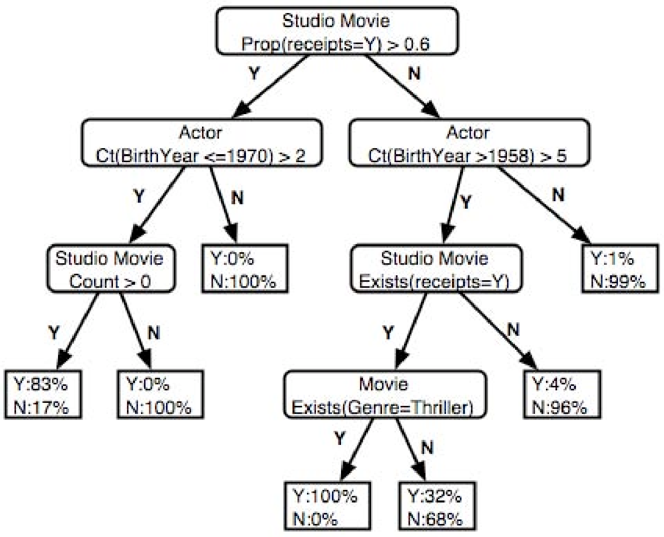


Fig. 4. Example RPT learned for the IMDb dataset to predict movie *receipts* given the labels of related movies.

Inference. As in the case of DNs, we use Gibbs sampling for inference in RDN models. For classification tasks, we want to estimate the full joint distribution over the target attributes in the graph. During inference, the model graph consists of a rolled-out network with both observed and unobserved variables. For example, we may want to use the network in Figure 3 to predict *movie.receipts* given *studio.country*, *actor.gender* and *actor.hasAward*. In this case, all attribute values are observed except *movie.receipts*. The values of the target variable are initialized to values drawn from the prior distribution (e.g. default distribution of *movie.receipts* in the training set). Gibbs sampling then proceeds iteratively, estimating the full joint distribution in cycles. For each target variable, the RPT model is used to return a probability distribution given the current attribute values in the rest of the graph. A new value is drawn from the distribution, assigned to the variable and recorded. This process is repeated for each unobserved variable in the graph. After a sufficient number of iterations, the values will be drawn from a stationary distribution and we can use the samples to estimate the full joint distribution. There are many implementation issues that could improve the estimates obtained from a Gibbs sampling chain, such as length of "burn in" and number of samples. We have not yet investigated the effects of these decisions on RDN performance. For the experiments reported in this paper we do not use a burn-in period and we use a fixed length chain of 2000 samples.

4 Experiments

This paper focuses on evaluation of RDNs in a classification context, where only a single attribute is unobserved in the test set—others are assumed to be observed and are not modeled with CPDs.

The experiments reported below are intended to evaluate two assertions. The first claim is that dependencies among instances can be used to improve model accuracy. We evaluate this claim by comparing the performance of two models. The first model is a conventional RPT model—an individual classification model that does not use labels of related instances, reasoning about each instance independently from other instances. We call this model *RPT-indiv*. The second model is a collective classification RDN model that exploits additional information available in labels of related instances and reasons about networks of instances collectively.

The second claim is that the RDN models, using Gibbs sampling, can effectively infer labels for a network of instances. To evaluate this claim, we include two more models for comparison. The third model is a conventional RPT model in which we allow the true labels of related instances to be used during both learning and inference. We call this model *RPT-ceiling*. This model is included as a ceiling comparison for the RDN model. It shows the level of performance possible if the model knew the *true* labels of related instances. The fourth model is intended to assess the need for a collective inference procedure. We call this model *RPT-default*. It reports the performance achieved on the first round of Gibbs sampling. This is equivalent to learning a conventional RPT model with the true labels of related instances, then randomly assigning labels according to the prior distribution of labels to use for inference. Since the test sets have connections to labeled instances in the training set (see next section for details), it is possible that these labels could provide enough information for accurate inferences, making Gibbs sampling unnecessary.

4.1 Tasks

The first data set is drawn from the Internet Movie Database (IMDb) (www.imdb.com). We gathered a sample of all movies released in the United States from 1996 through 2000, with opening weekend receipt information. The resulting collection contains 1383 movies. In addition to movies, the data set contains associated actors, directors, producers, and studios. In total, the data set contains 46,000 objects and 68,000 links. The learning task was to predict movie opening-weekend box office receipts. We discretized the attribute so that a positive label indicates a movie that garnered more than $2 million in opening-weekend box office receipts (*receipts*) (*P(Y)=0.45*).

We created training set/test set splits by temporal sampling into five sets, one for each year. Links to the future were removed from each sample. For example, the sample for 1997 contains links to movies from 1996, but not vice versa. We trained on the movies from one year (e.g. 1996) and tested on movies from the following year (e.g. 1997). Notice that during inference, there are links from the test set to fully labeled instances in the training set. This approach to sampling is intended to reproduce the expected domain of application for these models.

The RPT learning algorithm was applied to subgraphs centered on movies. The subgraphs consisted of movies and everything up to two links away—actors, directors, producers and studios, as well as movies associated with those objects. Nine attributes were supplied to the models, including studio country, actor birth-year and the class label of related movies two links away. Figure 4 shows an example RPT learned with the class labels of related movies. For a given movie *x*, the objects tagged "Studio Movie" refer to the other movies made by the primary studio associated with *x*.

The second data set is drawn from Cora, a database of computer science research papers extracted automatically from the web using machine learning techniques (McCallum, Nigam, Rennie and Seymore 1999). We selected the set of 1478 machine-learning papers published from 1994 through 1998, along with associated authors, journals, books, publishers, institutions and references. The resulting collection contains 11,500 objects and 26,000 links. The prediction task was to identify paper topic. Machine learning papers are divided into seven topics {Reinforcement Learning, Case-Based Reasoning, Probabilistic Methods, Theory, Genetic Algorithms, Neural Networks, and Rule Learning}.

As with the IMDb, we created training set/test set splits by temporal sampling into five sets, one for each year. The RPT learning algorithm used subgraphs centered on papers. The subgraphs consisted of papers, authors, journals, books, publishers, institutions and references, as well as papers associated with the authors. Twelve attributes were available to the models, including the journal affiliation, paper venue, and the topic of papers one link away (references) and two links away (through authors). Figure 5 shows an example RPT learned with the topics of related papers.

The RPT learning algorithm used randomization tests for feature selection and a Bonferroni-adjusted p-value growth cutoff of □=0.05. The RDN algorithm used a fixed number of Gibbs sampling iterations (2000).

4.2 Results and Discussion

Table 1 shows accuracy results for each of the four models on the IMDb classification tasks. On the IMDb, the RDNs models perform comparably to the *RPT-ceiling* models. This indicates that the RDN model realized its full potential, reaching the same level of performance as if it had access to the *true* labels of related movies.

In addition, the performance of the RDN models is superior to both the *RPT-indiv* models (RPT learned without labels) and the *RPT-default* models (RPT learned with labels and tested with a default labeling for the class values of related instances). In the example RPT in Figure 4 we can see that two of the five features refer to the target label on related movies. This indicates that autocorrelation is both present in the data and identified by the RPT models. The performance improvement over *RPT-indiv* is due to successful exploitation of this autocorrelation.

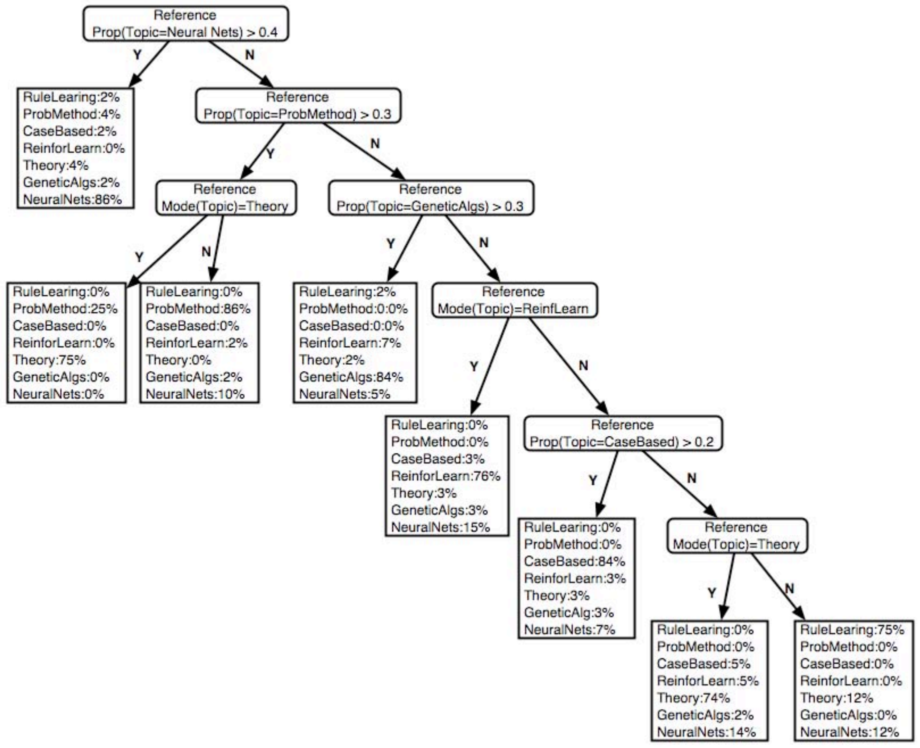


Fig. 5. Example RPT learned for the Cora dataset to predict paper topic given the topics of related papers.

We used two-tailed, paired t-tests to assess the significance of the accuracy results obtained from the four trials. The t-tests compare the RDN results to each of the three other models. The null hypothesis is that there is no difference in the accuracies of the two models; the alternative is that there is a difference. The resulting p-values are reported in the bottom row of the table, in the column of the model being compared to the RDN. The results support our conclusions above that the RDN results are significantly better than both *RPT-indiv* and *RPT-default*. Although the average performance of the RDNs is slightly lower than the *RPT-ceiling* models, the t-test indicates that this difference is not significant.

Table 1: Accuracy results for IMDb task

	RPT-indiv	*RPT-ceiling*	*RPT-default*	*RDN*
1	0.7148	0.8303	0.7473	0.8628
2	0.7500	0.8052	0.7370	0.8084
3	0.6893	0.8357	0.7429	0.7857
4	0.7103	0.8318	0.7383	0.8224
Avg	0.7161	0.8258	0.7414	0.8198
t-Test	0.0113	0.7529	0.0137	

On the Cora classification task, shown in Table 2, the RDN models show significant gains over the *RPT-indiv* and *RPT-default* models. This indicates that most of the predictive information lies in the topics of related pages. Nearly 90% of the features selected for the trees involved the topic of referenced papers. (Recall that reference topic was one of twelve attributes available to the model to form features.)

In this experiment, RDN models did not achieve the level of performance possible if the true label of related papers were known. However, the improvement from *RPT-indiv* and *RPT-default* models is notable. This is due to the paucity of predictive attributes other than the target label, clearly showing how autocorrelation can be exploited to improve model accuracy.

Table 2: Accuracy results for Cora task

	RPT-indiv	*RPT-ceiling*	*RPT-default*	*RDN*
1	0.2813	0.7437	0.4429	0.6852
2	0.2456	0.7646	0.4429	0.7013
3	0.2619	0.8027	0.5374	0.7313
4	0.2689	0.8571	0.5630	0.7983
Avg	0.2644	0.7920	0.4966	0.7290
t-Test	0.0004	0.0002	0.0004	

The difference in accuracy between the RDN and *RPT-ceiling* models may indicate that Gibbs sampling had not converged within the 2000 trials. To investigate this possibility we tracked accuracy throughout the Gibbs sampling procedure. Figure 6 shows curves for each of the tasks based on number of Gibbs iterations. The four lines represent each of the trials. Although we used 2000 iterations, we limit the graph because the accuracy plateaus within the first 150 iterations. Accuracy improves very quickly, leveling within the first 50 iterations. This shows that the approximate inference techniques employed by the RDN may be quite efficient to use in practice. We are currently experimenting with longer Gibbs chains, random restarts and convergence metrics to fully assess this aspect of the model.

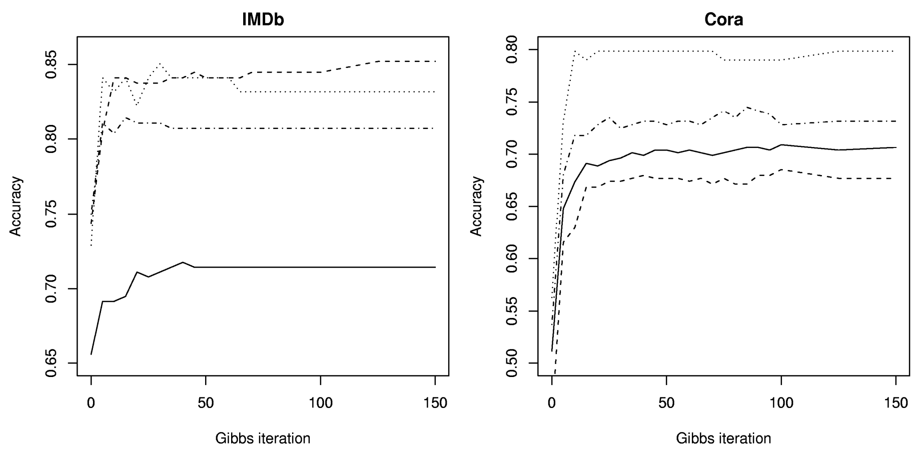


Fig. 6. Accuracy vs. number of Gibbs iterations. Each curve represents a separate trial.

If we can conclude from the learning curves that the Gibbs chain had converged, why didn't the RDN model perform as well as the *RPT-ceiling* model on Cora? One possible explanation is the lack of predictive attributes other than *topic*. The Gibbs chain may mix slowly, making the procedure unlikely to jump to distant portion of the labeling space. The inference procedure will suffer when there are no predictive attributes known with certainty to drive the mixing process in the right direction.

5 Conclusions and Future Work

These results indicate that collective classification with RDNs can offer significant improvement over non-collective approaches to classification when autocorrelation is present in the data. The performance of RDNs can approach the performance that would be possible if all the class labels of related instances were known. In addition, RDNs offer a relatively simple method for learning the structure and parameters of a graphical model, and they allow us to exploit existing techniques for learning conditional probability distributions. Here we have chosen to exploit our prior work on RPTs, which construct particularly parsimonious models of relational data, but we expect that the general properties of RDNs would be retained if other approaches to learning conditional probability distributions were used, given that those approaches are both selective and accurate.

Acknowledgments

Andrew McCallum provided valuable comments and suggestions on this work. We thank Michael Hay and Pippin Wolfe for comments. This research is supported under a National Science Foundation Graduate Research Fellowship and by DARPA, NSF and AFRL under contract numbers F30602-00-2-0597, EIA9983215 and F30602-01-2-0566. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements either expressed or implied, of the DARPA, the AFRL or the U.S. Government.

References

Bernstein, A., S. Clearwater, and F. Provost (2003). The relational vector-space model and industry classification. CDeR working paper #IS-03-02, Stern School of Business, New York University.

Chakrabarti, S., B. Dom and P. Indyk (1998). Enhanced hypertext categorization using hyperlinks. *Proc of ACM SIGMOD98*, pp 307-318.

Cortes, C., D. Pregibon, and C. Volinsky (2001). Communities of interest. *Proceedings Intelligent Data Analysis 2001.*

Della Pietra, S. V. Della Pietra and J. Lafferty (1997). Inducing features of random fields. I*EEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4): 380-393.

Domingos, P., M. Richardson. Mining the Network Value of Customers (2001). *Proceedings of the 7th International Conference on Knowledge Discovery and Data Mining*, pp. 57-66.

Getoor, L., N. Friedman, D. Koller, and A. Pfeffer (2001). Learning probabilistic relational models. In *Relational Data Mining*, Dzeroski and Lavrac, Eds., Springer-Verlag.

Heckerman, D., D. Chickering, C. Meek, R. Rounthwaite, and C. Kadie (2000). Dependency networks for inference, collaborative filtering and data visualization. *JMLR*, 1:49--75.

Jensen, D. and J. Neville (2002). Linkage and autocorrelation cause bias in relational feature selection. *Machine Learning: Proceedings of the Nineteenth International Conference*. Morgan Kaufmann.

Jensen, D., J. Neville and M. Hay (2003). Avoiding bias when aggregating relational data with degree disparity. *Proc. of the 20th Intl Joint Conf. on Machine Learning*, to appear.

Kleinberg, J. (1999). Authoritative sources in a hyper-linked environment. *Journal of the ACM* 46:604-632.

Lafferty, J., A. McCallum and F. Pereira (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proceedings of ICML2001.*

McCallum, A., K. Nigam, J. Rennie and K. Seymore (1999). A Machine Learning Approach to Building Domain-specific Search Engines. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pp. 662-667.

Murphy, K., Y. Weiss, and M. Jordan (1999). Loopy belief propagation for approximate inference: an empirical study. *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence*.

Neal, R. (1993). Probabilistic inference using Markov chain Monte Carlo methods. Tech report CRG-TR-93-1, Dept of Computer Science, University of Toronto.

Neville, J. and D. Jensen (2000). Iterative Classification in Relational Data. *AAAI Workshop on Learning Statistical Models from Relational Data*, 42-49.

Neville, J., D. Jensen, L. Friedland, M. Hay (2003). Learning relational probability trees. *Proceedings of the 9th International Conference on Knowledge Discovery & Data Mining*, to appear.

Taskar, B., P. Abbeel and D. Koller (2002). Discriminative probabilistic models for relational data. *Proceedings of UAI2002.*

Simple Estimators for Relational Bayesian Classifiers

Jennifer Neville, David Jensen and Brian Gallagher
Knowledge Discovery Laboratory, Department of Computer Science,
University of Massachusetts Amherst, 140 Governors Drive, Amherst, MA 01003 USA
{jneville | jensen | bgallag}@cs.umass.edu

Abstract

In this paper we present the Relational Bayesian Classifier (RBC), a modification of the Simple Bayesian Classifier (SBC) for relational data. There exist several Bayesian classifiers that learn predictive models of relational data, but each uses a different estimation technique for modeling heterogeneous sets of attribute values. The effects of data characteristics on estimation have not been explored. We consider four simple estimation techniques and evaluate them on three real-world data sets. The estimator that assumes each multiset value is independently drawn from the same distribution (INDEPVAL) achieves the best empirical results. We examine bias and variance tradeoffs over a range of data sets and show that INDEPVAL's ability to model more multiset information results in lower bias estimates and contributes to its superior performance.

1. Introduction

This paper presents a modification of the Simple Bayesian Classifier (SBC) for relational data. The power of relational data lies in combining intrinsic information about objects in isolation with information about related objects and the connections between those objects. However, the data often have irregular structures and complex dependencies, which contradict the assumptions of conventional modeling techniques. In particular, the heterogeneous structure of relational data precludes direct application of a SBC model, which operates on attribute-value data. We consider several approaches to modeling data with a relational Bayesian classifier (RBC) and evaluate performance on three data sets. The approach that follows the spirit of SBC and assumes conditional attribute value independence appears to work best. (See [9] for an expanded version of this paper.)

The simplicity of the SBC stems from its assumption that attributes are independent given the class—an assumption rarely met in practice. Research investigating the effects of this assumption on performance has helped to better understand the range of applicability of the SBC. For example, Domingos and Pazzani [2] showed that the SBC performs well under zero-one loss even when the independence assumption is violated by a wide margin. This paper studies similar questions for relational data. We empirically evaluate four different techniques on several real-world data sets. We explore the techniques on simulated data sets, decomposing loss into bias and variance estimates [1]. Our experiments show that characteristics of relational data can bias certain estimators and that using estimators with decreased bias improves model performance.

2. Modeling Relational Data

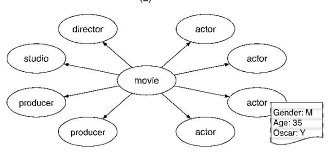
Relational data violate two assumptions of conventional classification techniques. First, algorithms for propositional data assume that the data instances are recorded in homogeneous structures (e.g. a fixed set of fields for each object), but relational data "instances" are consist of sets of heterogeneous records. Second, algorithms for propositional data assume that the data instances are independent and identically distributed (i.i.d.), but relational data have dependencies both through direct relations and through chaining multiple relations together. In this paper, we evaluate simple algorithms for learning models of data sets with heterogeneous instances. We do not attempt to exploit dependencies among related instances.

Relational data often have complex structures that are more difficult to model than homogeneous instances. For example, in order to predict the box-office success of a movie, a relational model might consider not only the attributes of the movie, but also attributes of the movie's actors, director, producers, and the studio that made the movie. A model might even consider attributes of indirectly related objects such as other movies made by the director. Each movie may have a different number of related objects, resulting in diverse structures. For example, some movies may have 10 actors and others may have 1000. When trying to predict the value of an attribute based on the attributes of related objects, a relational classification technique must consider *multisets* of attribute values. For example, we might model the likelihood of movie success given the multiset of gender values from the movie's actors.

There are a number of approaches to modeling sets of attribute values. *Propositionalization* is a common technique to transform heterogeneous data instances into homogenous records, mapping sets of values into single values with aggregation functions. A second approach is to treat the set of values independently and aggregate the resulting probability distributions using combining rules such as *noisy-or* or *average* [4]. A third approach is to model the sets directly with multinomials [7] or complex set-valued estimators [6].

This paper considers four estimation techniques from the range of approaches outlined above. Recent work has demonstrated the feasibility of these approaches for statistical models of relational data, but the choice of technique for any one model has been approached in a relatively ad-hoc manner. A thorough understanding of the effects of relational data characteristics on estimator performance will improve parameter estimation for relational data and should inform the development of more complex statistical models.

(a)



(b)

Receipts >$2mil	Actor Gender	Actor Age	Actor Oscar	Studio Location
+	{F,M,F,F}	{29,35, 32,18}	{N,N,N,Y}	{CA}
+	{M,M,F}	{31,45, 24}	{N,Y,N}	{NY}
-	{F,F}	{52,50}	{N,N}	{CA}
+	{M,M,F,F}	{24,27, 18,21}	{N,N,N,N}	{Canada}

Figure 1. Relational data represented as (a) a subgraph, and (b) decomposed by attribute.

3. Relational Bayesian Classifiers

The RBC represents heterogeneous examples as homogenous sets of attribute multisets. For example, a movie subgraph contains information about a number of related objects, such as actors and studios (e.g. Figure 1a). Transformed examples contain a multiset of values for each attribute, such as actor-age and studio-location (e.g. Figure 1b). This enables a SBC approach, where learning a model consists of estimating conditional probabilities for each attribute. However, estimation techniques for these data will need to model multisets of varying cardinality and high dimensionality. We refer to techniques used to estimate these probabilities as *estimators*. We will evaluate three approaches to estimation and four approaches to inference.

Average Value—The average value estimator (AVGVAL) corresponds to propositionalizing the data by averaging. During estimation, each multiset is replaced with its average value (for continuous attributes) or modal value (for discrete attributes). The average values are used in a standard maximum-likelihood estimator and probabilities are inferred from average/modal values as well. AVGVAL estimators are commonly used in probabilistic relational models (PRMs) to model dependencies where the "parent" consists of a set of attribute values [3]. We hypothesize that AVGVAL should perform well if the multiset values are highly correlated, so the multiset is no more informative than the average.

Random Value—The random value estimator (RANDVAL) is similar to AVGVAL. However, instead of deterministically choosing the most prevalent value from the set, RANDVAL chooses a representative value stochastically. This allows the estimation to differentiate between relatively uniform sets of values and highly skewed sets. This approach is equivalent to the *stochastic-mode* aggregation used in PRMs for classification [10]. Although RANDVAL may be more sensitive to the distribution of values in the sets, it may also experience greater variance if multiset values are distributed uniformly over a large range.

Independent Value—The independent value estimator (INDEPVAL) assumes each multiset value is independently drawn from the same distribution. This estimator is designed to mirror the independence assumption of SBC—now in addition to attribute independence, there is also an assumption of *attribute value* independence given the class. INDEPVAL models the multiset with a multinomial distribution where the size of the set is independent of the class. INDEPVAL should perform well if the multiset can be used to reduce variance, when there is little correlation among attribute values.

Average Probability—The fourth estimator (AVGPROB) aggregates probability distributions. It is an inference technique only (INDEPVAL is used for estimation). During inference, each multiset value's probability is computed independently and then the set of probabilities is averaged. This approach is one of the combining rules used in Bayesian logic programs (BLPs) to integrate probabilities into logic programs [4]. AVGPROB computes an arithmetic average of probabilities. If the set values are dependent, geometric averaging (used in INDEPVAL) will push the probabilities to extreme values. However, geometric averaging is more robust to irrelevant values, which pull arithmetic averages toward the center and wash out the effects of the useful values.

4. Empirical Data Experiments

The experiments reported below evaluate the claim that RBC models using INDEPVAL estimators will outperform RBC models using AVGVAL, RANDVAL or AVGPROB estimators. We compare the performance of each estimator on three real-world classification tasks. To compare the approaches, we recorded accuracy and area under the ROC curve using ten-fold cross-validation.

4.1. Classification Tasks

The first data set, drawn from the Internet Movie Database (IMDb) (www.imdb.com), is a sample of all movies released in the United States from 1996 to 2001, with opening weekend box-office receipt data. The sample contains 1383 movies and related actors, directors, producers, and studios. The task was to predict whether a movie made more than $2mil in opening weekend receipts (*P(+)=0.45*). Nine attributes were supplied to the models, including studio country and actor birth-year.

The second data set, drawn from Cora [8], is a sample of 4330 machine-learning papers and associated authors, journals/books, publishers, and cited papers. The task was to predict whether a paper's topic is *Neural Networks* (*P(+)=0.32*). Ten attributes were available to the models, including journal affiliation and paper venue.

The third data set contains information about 1243 genes in the yeast genome and 1734 interactions among their associated proteins (www.cs.wisc.edu/~dpage/kddcup2001/). The task was to predict whether a gene's functions include *Transcription* (*P(+)=0.31*). Fourteen attributes where supplied to the models, including gene phenotype, motif, and interaction type.

4.2. Results

Figure 2 shows AUC results for each of the models on the three classification tasks, averaged over the ten folds. Accuracy results are comparable [9]. We used two-tailed, paired t-tests to assess the significance of the ten-fold cross-validation results, comparing INDEPVAL to each of the other estimators. Asterisks in Figure 2 indicate a significant difference in performance compared to INDEPVAL (p-value < 0.001).

On the IMDb and Cora classification tasks, INDEPVAL's AUC results are superior to any of the other approaches. The performance of AVGVAL and RANDVAL indicates that propositionalizing relational data (even stochastically) to apply conventional models may not always be a good approach. On the Gene task, all approaches perform equivalently.

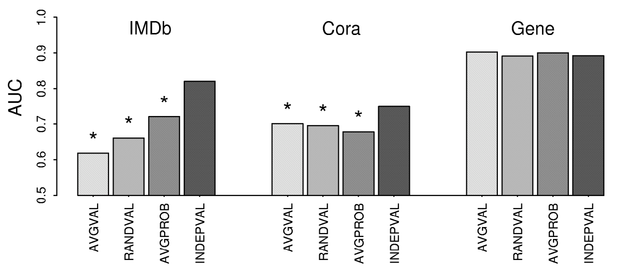


Figure 2: Results of empirical data experiments for IMDb, Cora, and Gene databases.

5. Synthetic Data Experiments

We use synthetic data to explore the effects of linkage, attribute correlation, and multiset distributions on estimator performance. Relational data sets often exhibit concentrated linkage, where certain object types have a large number of relations. For example, papers in Cora link to a few journals, and movies in the IMDb link to a small number of studios. Uniformity among attribute values of objects that share a common neighbor is also common in relational data. For example, in the gene data, proteins located in the same place in the cell often have highly correlated functions.

5.1. Methodology

Our synthetic data sets are comprised of bipartite graphs, each containing a single core object (e.g. a movie) linked to zero or more peripheral objects (e.g. actors). Note that each actor links to exactly one movie. Each movie has a binary class label, C={+,-}, and each actor has a binary attribute, A={1,0}. The number of actors per movie is distributed normally with mean equal to |actors|/|movies|. The default experimental parameters were 100 movies, 500 actors, *P(C=+)=0.5*, and *P(A=1|C=+)=P(A=0|C=-)=0.75*. Variations from these defaults are described for each experiment below.

We measured average zero-one loss and squared-loss for each RBC estimator across 100 pairs of training/test sets and decomposed loss into bias and variance [1]. Bias and variance estimates were calculated for each test example using 100 different training sets and averaged over the entire test set. This was repeated for 100 test sets and averaged. The zero-one loss results are presented in Figure 3. Squared-loss results are similar [9].

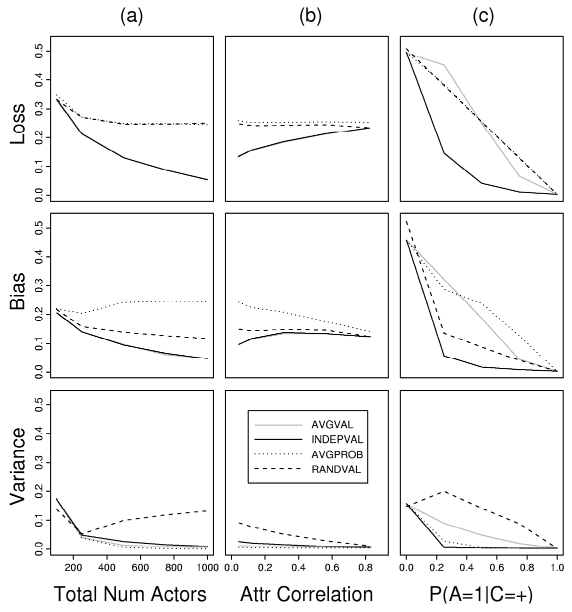


Figure 3: Results of synthetic data experiments.

5.2. Results

The experiment shown in Figure 3a varied the total number of actors in each data set from 100 to 1000. In this experiment AVGVAL and INDEPVAL are nearly indistinguishable, as are AVGPROB and RANDVAL. For all estimators except RANDVAL, increasing degree reduces variance. This was expected, as the variance of the random value selection increases with set size. AVGPROB's arithmetic averaging cannot exploit the extra information in larger sets, which results in higher bias.

The experiment in Figure 3b varied the correlation among linked actor attribute values from [0.05,0.85]. Again, AVGVAL and INDEPVAL are indistinguishable. As attribute correlation increases, the bias of the INDEPVAL estimator increases, indicating that INDEPVAL's probability estimates may be skewed in data with high attribute correlations.

The experiment in Figure 3c varied *P(A=1|C=+)* from [0,1] while holding *P(A=1|C=-)* constant at 0. This is the first experiment to show a difference between AVGVAL and INDEPVAL, illustrating performance when rare attribute values determine the class. Since INDEPVAL shows lower bias than either of the other estimators we can attribute its higher accuracy to this reduction in bias.

Given these results, the relative strength of INDEPVAL appears to lie in the estimator's ability to make use of rare attribute values, as well as multiple predictive values within a multiset. To determine if these types of multisets occur in practice, we examined multisets from the IMDb. We calculated the correlation of each attribute value with the class label using chi-square, assessed significance after adjusting for multiset size [5], and then determined the number of unique correlated attribute values per movie. Figure 4 shows the frequency distribution of these counts across movies for three example attributes. A large number of movie subgraphs have more than one unique attribute value correlated with the class. In this situation, estimators that can capture more multiset information (e.g. INDEPVAL) will outperform estimators that propositionalize to a single value (e.g. AVGVAL).

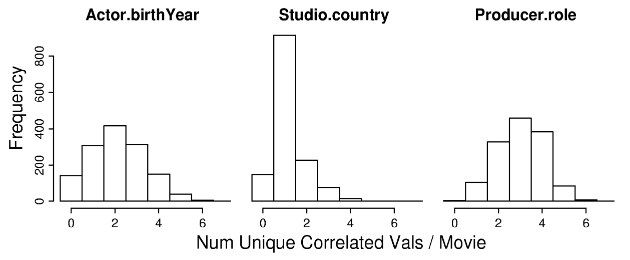


Figure 4: Count of unique significantly correlated values in each subgraph, for three attributes in the IMDb.

6. Conclusions

We have identified a simple approach to estimation for relational data. Adhering to the spirit of SBC simplicity, the RBC model that assumes conditional independence of both attributes and multiset attribute values (INDEPVAL) is successful in a variety of real-world classification tasks. This model is easy to implement and efficient to use, making it a good baseline for evaluation of more complex relational learning techniques.

INDEPVAL estimators have low bias and variance over a wide range of synthetic data sets. AVGVAL has low variance over a number of conditions, but it is easy to identify situations in which AVGVAL is a biased estimator. We can infer that INDEPVAL's superior performance on the real-world classification tasks is a result of lower overall bias—due to its ability to exploit information contained in both rare values and multiple correlated values within the sets. AVGPROB appears to be biased over a number of data sets, but it performs quite well on the IMDb. This reveals that our synthetic data experiments have not clearly identified the circumstances in which AVGPROB is a good approach to estimation.

Future work will include further analysis of the effects of relational data characteristics on complex multiset estimators [e.g. 6] and development of models that select attribute estimators based on data characteristics.

7. Acknowledgments

We thank Ross Fairgrieve for his contributions to an earlier draft of this work. This research is supported by DARPA and AFRL, AFMC, USAF under contract numbers F30602-00-2-0597 and F30602-01-2-0566.

8. References

[1] Domingos, P. A Unified Bias-Variance Decomposition for Zero-One and Squared Loss. *Proceedings of the 17th National Conference on Artificial Intelligence*, 2000.

[2] Domingos, P. and M. Pazzani. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29:103-130, 1997.

[3] Getoor, L., N. Friedman, D. Koller, and A. Pfeffer. Learning probabilistic relational models. *Relational Data Mining*, Dzeroski and Lavrac, Eds., Springer-Verlag, 2001.

[4] Kersting, K. and L. De Raedt. Basic principles of learning Bayesian logic programs. Tech Report 174, University of Freiburg, Germany, June 2002.

[5] Jensen, D., J. Neville and M. Hay. Avoiding bias when aggregating relational data with degree disparity. *Proc. of the 20th International Conf. on Machine Learning,* 2003.

[6] Lachiche, N. and P. Flach 1BC2: a true first-order Bayesian Classifier. *Proceedings of the 12th International Conference on Inductive Logic Programming,* 2002.

[7] McCallum, A. and K. Nigam. A comparison of Event Models for Naive Bayes Text Classification. In *AAAI-98 Workshop on Learning for Text Categorization*, 1998.

[8] McCallum, A., K. Nigam, J. Rennie and K. Seymore. A Machine Learning Approach to Building Domain-specific Search Engines. *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, 1999.

[9] Neville, J., D. Jensen and B. Gallagher. Simple Estimators for Relational Bayesian Classifiers. University of Massachusetts, Technical Report 03-04.

[10] Taskar, B., E. Segal, and D. Koller. Probabilistic Classification and Clustering in Relational Data. *Proceedings of the 17th Intl Joint Conference on Artificial Intelligence,* 2001.

Statistical Relational Learning: Four Claims and a Survey

Jennifer Neville, Matthew Rattigan, David Jensen
Knowledge Discovery Laboratory, Department of Computer Science, University of Massachusetts,
140 Governors Drive, Amherst, MA 01003 USA
{jneville | rattigan | jensen } @cs.umass.edu

Statistical relational learning (SRL) research has made significant progress over the last 5 years. We have successfully demonstrated the feasibility of a number of probabilistic models for relational data, including probabilistic relational models, Bayesian logic programs, and relational probability trees, and the interest in SRL is growing. However, in order to sustain and nurture the growth of SRL as a subfield we need to refocus our efforts on the science of machine learning — moving from demonstrations to comparative and ablation studies. We will outline four assertions that are implicit to SRL research but which have been only minimally evaluated. We hope to stimulate discussion as to how, as a community, these claims can be addressed in future research.

1 Introduction

In the hopes of generalizing the results of recent research from the statistical relational learning (SRL) community, we surveyed twenty recent SRL papers. From the papers studied we were able to distill four implicit claims that underlie much of the current SRL research. We present an examination of those claims in the context of the papers surveyed.

We chose twelve of the papers as a representative sample for the purposes of this discussion. Each paper chosen describes and evaluates a discriminative, probabilistic relational model. A descriptive list of the selected models and papers appears in Table 1.

The purpose of this paper is to stimulate a discussion of the scientific methods that will help to illustrate and evaluate the relative merits of the different models and their frameworks.

2 Relational vs. propositional

Claim: Models learned from both intrinsic and relational information perform better than those learned from intrinsic information alone, and are therefore worth the added complexity.

This is an implicit claim of relational learning in general. We expect that predictive information exists in relationships among instances, and that this information can be used to reduce model bias. However, decreasing bias often results in increased variance (Friedman 1997). This is a very real concern for relational learning algorithms that are faced with an exponential explosion in the size of the model space.

The simplest way to evaluate this claim is to record model performance using *intrinsic* data, a subset of the data where relational information is removed. By this we mean data where the instances are objects in isolation, and the only information available are the attributes intrinsic to those objects as individuals. Popescul, Ungar, Lawrence, and Pennock (2003) use this approach when evaluating their models on citation data, comparing models learned on information intrinsic to documents alone with those learned from both intrinsic and citation information. Getoor, Segal, Taskar and Koller (2001) use an alternative approach, including results from a baseline propositional model learned on intrinsic data. This technique is also employed in four other papers. See figure 2 for details.

More than half of the papers surveyed included some comparative intrinsic analysis, and the results vary considerably across models and datasets. For example, when using relational features Neville, Jensen, Gallagher, and Fairgrieve (2003) found marked improvement in model performance on two datasets, but no significant gain on a third. We believe that this type of analysis is important baseline for determining whether the inclusion of relational information is of any benefit, and if so whether the additional model complexity is warranted. Although preliminary analysis along these lines is a common component of SRL research, we feel that more explicit and directed experimentation is needed to fully justify the use SRL models for relational datasets.

3 Probabilistic vs. deterministic

Claim: Probabilistic relational models offer significant advantages over deterministic relational models in relational domains.

Table 1: Statistical relational learning models surveyed

Model	Description	Selective	Generative	Reference
RVS	relational vector-space model	No	no	Bernstein, Clearwater, and Provost, 2003
FOIL-PILFS	relational learner w/statistical predicate invention	Yes	no	Craven and Slattery, 2001
Maccent	maximum entropy model with clausal constraints	Yes	no	Dehaspe, 1997
SNM	Markov random field for social networks	No	no	Domingos and Richardson, 2001
BLP	Bayesian logic programs	yes	yes	Kersting and De Raedt, 2002
1BC2	first-order naive Bayesian classifier	no	no	Lachiche and Flach, 2002
RBC	relational Bayes classifier	no	no	Neville, Jensen, Gallagher and Fairgrieve, 2003
RPT	relational probability trees	yes	no	Neville, Jensen, Friedland and Hay, 2003
SLR	structural logistic regression	yes	no	Popescul, Ungar, Lawrence, and Pennock, 2003
NBILP-R	naive Bayes classifier with ILP features	no	no	Pompe and Kononenko, 1995
PRM	probabilistic relational model	yes	yes	Getoor, Segal, Taskar and Koller, 2001
RMN	relational Markov network	no	no	Taskar, Abbeel and Koller, 2002

Research in relational learning has investigated deterministic models for many years (e.g. Muggleton & De Raedt 1994, Lavrac & Dzeroski 1994). Recent efforts have shifted the focus towards a probabilistic setting. We outline a number of advantages of probabilistic models below, but we feel that discussion of the strengths and weaknesses of each technique is worth exploring in greater detail. Discussion along these lines is necessary to come to a general understanding of the range and applicability of SRL models.

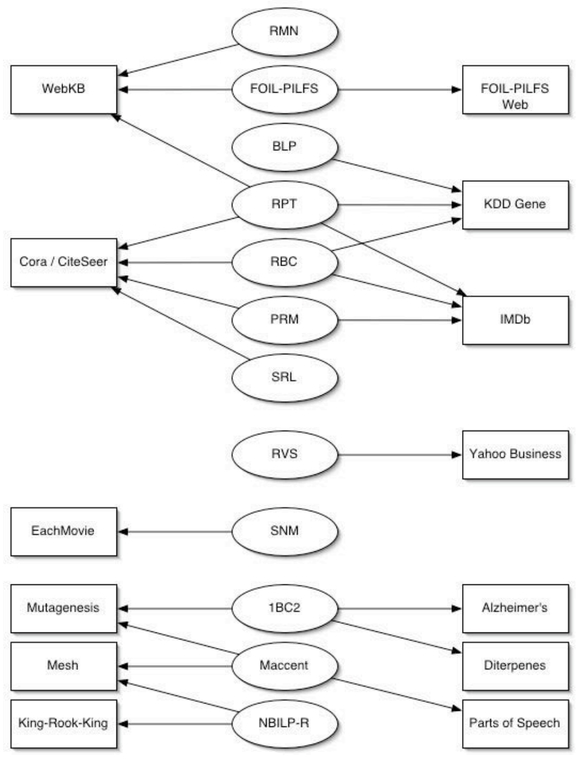


Figure 1: SRL models and evaluation datasets.

One strength of probabilistic models is the ability to evaluate how these models will perform over a range of class and cost distributions (Provost and Fawcett, 1997). Classification tasks involving complex relational data often have varying levels of misclassification costs as well as uncertain class distributions. Since deterministic models do not associate a level of confidence with their classifications, it is difficult to estimate their behavior in these domains.

Another advantage of probabilistic models is their suitability to real-world analysis tasks. Since these models generate meaningful, continuous probability scores, they lend themselves to an iterative, hierarchical approach to analysis. As Bernstein, Clearwater, and Provost (2003) point out, "scores may be most useful as feature constructors in other, more complicated systems." It is therefore crucial to evaluate the probabilities produced in SRL models quantitatively; unfortunately, none of the papers we surveyed perform this type of evaluation. Secondly, probability scores allow us to rank instances in order of certainty. This is of great use to real-world analysts who have limited time to investigate "positive" instances, as confidence scores allow an analyst to prioritize instances rather than treat all members of a predicted class equally.

Finally, probabilistic models are in general more suited to learning with relational data than deterministic ones. Due to their complexity, relational datasets are often noisy, which can be troublesome for deterministic models (Popescul et al. 2003). Furthermore, the advantage of working with relational data may be lost without the use of probabilistic models. For example, Craven and Slattery (2001) found in the text classification domain that "learned rules will not be dependent on the presence or absence of specific key words as a conventional relational method. Instead, the statistical classifiers in its learned rules consider the weighted evidence of many words."

4 Heterogeneous data

Claim: SRL algorithms learn accurate models of structured data.

Most conventional classification techniques assume data instances are recorded in homogeneous structures. Relational data however, often have complex structures that are difficult to model in propositional form. For example, information about actors, directors and producers may be useful when building a model of movie success but each movie has a different number of related entities. This variety results in examples with diverse structure — some movies may have 10 actors, and others may have hundreds. The ability to deal with heterogeneous data instances is a defining characteristic of relational learning algorithms.

The relational learning community has developed a number of models that can handle heterogeneous data. For example, Lachiche and Flach (2002) extend conventional naive Bayes classifiers to handle heterogeneous instances and Deshape (1997) extends conventional maximum entropy models to use a richer first-order logic format.
Each of the 12 papers surveyed introduces a different model for this purpose. However, few of these papers evaluate the effects of heterogeneity on the learned models. Some of our recent work has examined how particular characteristics of relational data affect the statistical inferences necessary for accurate learning (Jensen & Neville 2002, Jensen, Neville & Hay 2003). Specifically, we have shown that concentrated linkage combined with high autocorrelation can lead to feature selection bias if models are constructed naively. Also, we have shown that degree disparity can lead to spurious correlations in aggregated features, resulting in overly complex models with excess structures.

These characteristics of relational data can greatly complicate efforts to construct good statistical models. Only selective models are vulnerable to the particular biases mentioned above, but 7 of the models surveyed do some form of selection while learning. It is difficult to evaluate models for unidentified biases; however, comparative studies among the various SRL algorithms should help to uncover these biases. In particular, detailed comparisons of selective and non-selective model performance may help to uncover additional biases. Figure 2 depicts the 12 SRL models with links to the various models compared to during evaluation. The paucity of outlinks speaks for itself.

We have only begun to explore the effects of data characteristics on model learning. While many relational models outperform propositional models on the same datasets, the relational models may not be living up to their full potential. Further investigation of the complexities of relational data will help to identify sources of potential bias and correcting for these biases will unleash the full power of SRL models.

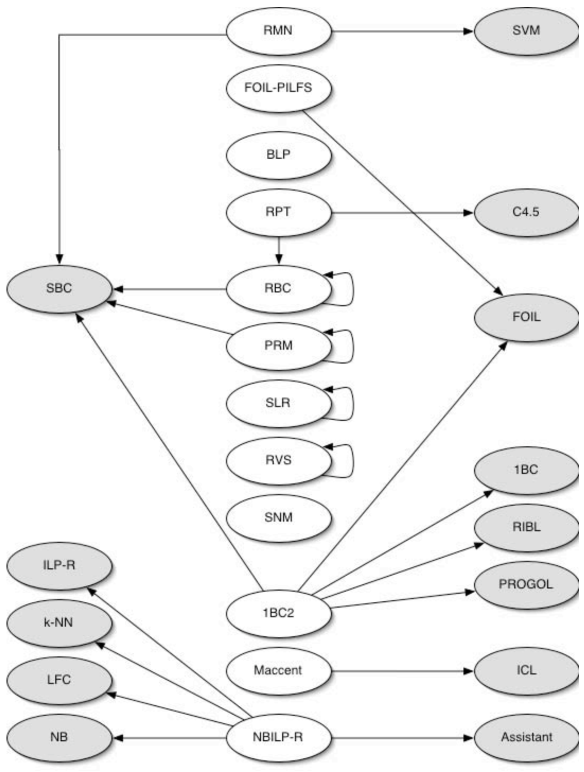


Figure 2: SRL models and evaluation models. Self-loops indicate ablation comparisons.

5 Interdependent data

Claim: SRL algorithms learn accurate models of dependent data instances.

Independence of instances is a deeply buried assumption of traditional machine learning methods that is contradicted by many relational datasets. For example, in scientific literature datasets there are dependencies among papers written by the same author and in web datasets there are dependencies among pages linked to by the same document. The structure of complex relational data such as these presents a unique opportunity for improving the accuracy of statistical models. If two objects are related, inferring something about one object can aid inferences about the other.

In our analysis of relational data, we have encountered many examples of dependencies that could be exploited to improve learning. For example, in analysis of the 2001 KDD Cup data we found that the proteins located in the same place in a cell (e.g., mitochondria or cell wall) had highly autocorrelated functions (e.g., transcription or cell growth). Such autocorrelation has been identified in other domains as well. For example, fraud in mobile phone networks has been found to be highly autocorrelated (Cortes, Pregibon & Volinsky 2001). The topics of authoritative web pages are highly autocorrelated when linked through directory pages that serve as "hubs" (Kleinberg 2001).

Table 2: Characteristics of data and sampling approach used for evaluation

Model	Data connectivity	Sampling approach
RVS	one large connected component	not mentioned
FOIL-PILFS	disjoint large graphs	leave-one-graph-out cross validation
Maccent	disjoint small graphs	leave-k-graph-out cross validation
SNM	one large connected component	sample by time
BLP	one large connected component	transduction
1BC2	disjoint small graphs	leave-k-graph-out cross validation
RBC	one large connected component	subgraph sampling
RPT	one large connected component	subgraph sampling
SLR	one large connected component	transduction
NBILP-R	disjoint small graphs	leave-k-graph-out cross validation
PRM	large conn comp / disjoint graphs	transduction / leave-one-graph-out cv
RMN	disjoint large graphs	leave-one-graph-out cross validation

Many of the models surveyed do not attempt to exploit dependencies among relational instances. More than half of the algorithms are designed to learn models relational datasets with independent, heterogeneous instances (i.i.d. relational data) where any dependencies among instances are ignored.

Inductive logic programming (ILP) models have been capable of representing dependencies among instances for years, albeit only extreme (deterministic) dependencies (Lavrac & Dzeroski 1994). However, it is only recently that statistical models have been developed to exploit the dependencies in relational data. For example, Kersting and De Raedt (2002) combine ILP with Bayesian networks to integrate probabilities into logic programs and model the dependencies among proteins in a cell. Getoor et al. (2001) use probabilistic relational models (PRMs) to model the the dependencies among hyperlinked web pages. Taskar, Abbeel and Koller (2002) use conditional Markov networks to model the same domain. Domingos and Richardson (2001) represent market entities as social networks and develop Markov random field models to model the influence in purchasing patterns throughout the network. Bernstein, Clearwater and Provost (2003) outline a relational vector-space model that uses autocorrelation to identify the group membership of linked entities.

Statistical models capable of collective classification across a network of instances are a relatively new phenomenon. It is unclear how to effectively evaluate the performance of these models. In what context do we expect to be using these models in the real world? Will we be applying the model to a completely new graph or do we expect new instances to arrive temporally related to the existing (training set) graph. Answers to this question should help to develop sampling methods to get an unbiased estimate of model performance.

Furthermore, how should we sample from a large connected graph? Table 2 outlines the characteristics of datasets examined by each of the models along with the sampling approach that was chosen. There are four approaches to sampling currently in use; more work is needed to determine which of these approaches is appropriate for a particular learning task.

6 Conclusions

Although the SRL community has successfully demonstrated the feasibility of a number of probabilistic models for relational data, there is much work to be done in order to begin generalizing the range and applicability of the various models. We have presented four claims for discussion with the purpose of advancing the science of SRL as well as machine learning in general.

References

Bernstein, A., S. Clearwater, and F. Provost. The relational vector-space model and industry classification. CDeR working paper #IS-03-02, Stern School of Business, New York University, 2003.

Cortes, C., D. Pregibon, and C. Volinsky. Communities of Interest. *Proceedings of the Fourth International Symposium on Intelligent Data Analysis*, 2001.

Craven, M. and S. Slattery. Relational learning with statistical predicate invention: Better models for hypertext. Machine Learning Journal 43:97-119, 2001.

Dehaspe, L. Maximum entropy modeling with clausal constraints. In Proceedings of the 7th International Workshop on Inductive Logic Programming, pages 109-124. Springer-Verlag, 1997.

Domingos, P., M. Richardson. Mining the Network Value of Customers. Proceedings of the 7th International Conference on Knowledge Discovery and Data Mining, pp. 57-66, 2001.

Friedman, J. On bias, variance, 0/1-loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, 1:55-77, 1997.

Getoor, L., E. Segal, B. Taskar and D. Koller. Probabilistic Models of Text and Link Structure for Hypertext Classification. Proceedings of the IJCAI01 Workshop on Text Learning: Beyond Supervision, 2001.

Jensen, D., J. Neville and M. Hay. Avoiding bias when aggregating relational data with degree disparity. Proceedings of the *20th Int. Joint Conf. on Machine Learning*, to appear.

Kersting, K., L. De Raedt. Basic principles of learning Bayesian logic programs. Technical Report No. 174, Institute for Computer Science, University of Freiburg, Germany, June 2002.

Kleinberg, J. Authoritative sources in a hyperlinked environment. *Journal of the ACM* 46:604-632, 1999

Lachiche, N. and P. Flach. 1BC2: a True First-Order Bayesian Classifier. Proceedings of the 12th International Conference on Inductive Logic Programming, 2002.

Lavrac, N. and Dzeroski. *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood, 1994.

Muggleton, S. editor. *Inductive Logic Programming*. Academic Press, 1992.

Neville, J., D. Jensen, B. Gallagher, R. Fairgrieve. Simple Estimators for Relational Data. University of Massachusetts, Technical Report 03-04, 2003.

Neville, J., D. Jensen, L. Friedland, M. Hay. Learning relational probability trees. Proceedings of the 9th International Conference on Knowledge Discovery & Data Mining, to appear.

Popescul, A., L. Ungar, S. Lawrence, D. Pennock, Statistical relational learning for document mining. Submitted, 2003.

Pompe, U. and I. Kononenko. Naive Bayesian classifier within ILP-R. Proceedings of the 5th International Workshop on Inductive Logic Programming, pages 417-436, 1995.

Provost, F. and T. Fawcett. Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. Proceedings of the 3rd International Conference on Knowledge Discovery & Data Mining, pages 43-48, 1997.

Taskar, B., P. Abbeel and D. Koller. Discriminative probabilistic models for relational data. Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence, 2002.

Dependency Networks for Relational Data

Jennifer Neville, David Jensen
Computer Science Department
University of Massachusetts Amherst
Amherst, MA 01003
{jneville|jensen}@cs.umass.edu

Abstract

Instance independence is a critical assumption of traditional machine learning methods contradicted by many relational datasets. For example, in scientific literature datasets there are dependencies among the references of a paper. Recent work on graphical models for relational data has demonstrated significant performance gains for models that exploit the dependencies among instances. In this paper, we present relational dependency networks (RDNs), a new form of graphical model capable of reasoning with such dependencies in a relational setting. We describe the details of RDN models and outline their strengths, most notably the ability to learn and reason with cyclic relational dependencies. We present RDN models learned on a number of real-world datasets, and evaluate the models in a classification context, showing significant performance improvements. In addition, we use synthetic data to evaluate the quality of model learning and inference procedures.

1. Introduction

Relational data pose a number of challenges for model learning and inference. The data have complex dependencies, both as a result of direct relations (e.g., research paper references) and through chaining multiple relations together (e.g., papers published in the same journal). The data also have varying structure (e.g., papers have different numbers of authors, references and citations). Traditional graphical models such as Bayesian networks and Markov networks assume that data consist of independent and identically distributed instances, which makes it difficult to use these techniques to model relational data that consist of non-independent and heterogeneous instances. Recent research in relational learning has produced several novel types of graphical models to address this issue. Probabilistic relational models (PRMs)[1] (e.g. [5, 15, 14]) estimate joint probability distributions of relational data and have been evaluated successfully in several domains, including the World Wide Web, genomic structures, and scientific literature.

In this paper, we present relational dependency networks (RDNs), an extension of dependency networks [6] for relational data[2]. RDN models are a new form of PRM that offer several advantages over the comparable joint models—relational Bayesian networks (RBNs) [5] and relational Markov networks (RMNs) [15]. Specifically, the strengths of RDNs include: (1) an interpretable representation that facilitates knowledge discovery in relational data, (2) the ability to represent arbitrary cyclic dependencies, including relational autocorrelation, and (3) simple and efficient methods for learning both model structure and parameters.

Graphical models are an attractive modeling tool for knowledge discovery because they are a compact representation of the joint distribution of a set of variables, which allows key dependencies to be expressed and irrelevancies to be ignored [2]. The qualitative properties of the model are encoded in the *structure* of the graph, while the quantitative properties are specified by the *parameters* of the associated probability distributions. The models are often easy to interpret because the graph structure can be used to infer dependencies among variables of interest. PRMs maintain this property as they extend conventional graphical models to relational settings. A compact representation is even more desirable for modeling relational data, because the enormous space of possible dependencies can overwhelm efforts to identify novel, and interesting patterns.

The ability to represent, and reason with, arbitrary cyclic dependencies is another important characteristic of relational models. Relational autocorrelation, a statistical dependency among values of the same variable on related entities [7], is a nearly ubiquitous phenomenon in relational datasets. For example, hyperlinked web pages are more

[1]Several previous papers (e.g., [5]) use the term PRM to refer to a specific type of model that the originators now call a relational Bayesian network (Koller, personal communication). In this paper, we use PRM in its more recent and general sense.

[2]This work generalizes our previous work on simple RDNs for classification [12].

likely to share the same topic than randomly selected pages, and proteins located in the same place in a cell are more likely to share the same function than randomly selected proteins. Recent work has shown that autocorrelation dependencies can be exploited to improve classification accuracy, if inferences about related data instances are made simultaneously (e.g., [3, 15, 12]). However, these relational autocorrelation dependencies are often cyclic in nature, making it difficult to encode these dependencies with directed graphical models such as RBNs unless the autocorrelation can be structured to be acyclic (e.g., with temporal constraints) [5]. In contrast, undirected graphical models such as RMNs, and RDNs, can represent arbitrary forms of relational autocorrelation.

During learning, relational models consider a large number of features, thus simple and efficient learning techniques are advantageous, particularly for joint models. The RDN learning algorithm is based on pseudolikehood techniques [1], which estimate a set of conditional distributions independently. This approach avoids the complexities of estimating a full joint distribution and can incorporate existing techniques for learning conditional probability distributions of relational data. Relatively efficient techniques exist for learning both the structure and parameters of RBN models but the acyclicity constraints of the model precludes the learning of arbitrary autocorrelation dependencies. On the other hand, while in principle it is possible for RMN techniques to learn cyclic autocorrelation dependencies, inefficiencies due to modeling the full joint distribution make this difficult in practice. The current implementation of RMNs is not capable of learning model structure automatically, nor can it automatically identify which features are most relevant to the task; research has focused primarily on parameter estimation and inference procedures. To our knowledge, RDNs are the first PRM capable of *learning* cyclic autocorrelation dependencies.

We begin by reviewing the details of dependency networks for propositional data and then describe how to extend them to a relational setting. Next, we present RDN models learned from four real-world datasets and evaluate the models in a classification context, demonstrating equivalent, or better, performance in comparison to conditional models. Finally, we report experiments on synthetic data that show model learning and inference is robust to varying levels of autocorrelation and that accurate models can be learned from small training set sizes.

2. Dependency Networks

Graphical models represent a joint distribution over a set of variables. The primary distinction between Bayesian networks, Markov networks and dependency networks (DNs) [6] is that dependency networks are an approximate representation. DNs approximate the joint distribution with a set of conditional probability distributions (CPDs), which are learned independently. This approach to learning is a relatively simple technique for parameter estimation and structure learning that results in significant efficiency gains over exact models. However, because the CPDs are learned independently, DN models are not guaranteed to specify a *consistent* joint distribution. This precludes DNs from being used to infer causal relationships and limits the applicability of exact inference techniques. Nevertheless, DNs can encode predictive relationships (i.e. dependence and independence) and Gibbs sampling (e.g. [11]) inference techniques can be used to recover a full joint distribution, regardless of the consistency of the local CPDs.

DN Representation. A DN encodes probabilistic relationships among a set of variables in a similar manner to Bayesian and Markov networks, combining characteristics of both undirected and directed models. DN models consists of a graph G, which encodes the *structure* of the model, and a set of probability distributions P, which encode the *parameters* of the model. Dependencies among variables are represented with a bidirected graph $G = (V, E)$, where conditional independence is interpreted using graph separation, as with undirected models. However, as with directed models, dependencies are quantified with a set of conditional probability distributions P. Consider the set of variables $\mathbf{X} = (X_1, ..., X_n)$ over which we'd like to model the joint distribution $p(\mathbf{x}) = p(x_1, ..., x_n)$. Each node $v_i \in V$ corresponds to an $X_i \in \mathbf{X}$ and is associated with a probability distribution conditioned on the other variables, $P(v_i) = p(x_i|\mathbf{x} - \{x_i\})$. The parents, pa_i, of node i are the set of variables that render X_i conditionally independent of the other variables ($p(x_i|pa_i) = p(x_i|\mathbf{x} - \{x_i\})$) and G contains a directed edge from each parent node v_j to each child node v_i ($e(v_j, v_i) \in E$ iff $x_j \in pa_i$).

DN Learning. Both the structure and parameters of DN models are determined through learning a set of local conditional probability distributions (CPDs). The DN learning algorithm learns a separate CPD for each variable X_i, conditioned on the other variables in the data ($\mathbf{X} - \{X_i\}$). Any conditional learner can be used for this task (e.g. logistic regression, decision trees). The learned CPD is included in the model as $P(v_i)$, and the variables selected by the conditional learner (e.g., $x_i = \alpha x_j + \beta x_k$) form the parents of X_i (e.g., $pa_i = \{x_j, x_k\}$), which is then reflected in the edges of G appropriately. If the conditional learner is not selective, the DN model will be fully connected (i.e., $pa_i = \mathbf{x} - \{x_i\}$). In order to build understandable DNs, it is desirable to use a selective learner that will learn CPDs that use a subset of all variables.

DN Inference. Although the DN approach to structure learning is simple and efficient, it can result in an inconsistent network, both structurally and numerically. In other words, there may be no joint distribution from which each of the CPDs can be obtained using the rules of probability. For example, a network that contains a directed edge from X_i to X_j, but not from X_j to X_i, is inconsistent—X_i and X_j are dependent but X_j is not represented in the CPD for X_i. A DN is consistent if the conditional distributions in P factor the joint distribution—in this case we can compute the joint probability for a set of values $\mathbf{x}$ directly. In practice, [6] show that DNs will be nearly consistent if learned from large data sets, since the data serve a coordinating function that ensures consistency among the CPDs. If a DN is inconsistent, approximate inference techniques can be used to estimate the full joint distribution and extract probabilities of interest. Gibbs sampling (e.g., [11]) can be used to recover a full joint distribution for $\mathbf{X}$, regardless of the consistency of the local CPDs, provided that each X_i is discrete and each local CPD is positive [6].

3. Relational Dependency Networks

RDNs extend dependency networks to a relational setting. DNs have been shown to perform comparably to BNs for a number of propositional tasks [6], thus we expect they will achieve similar performance levels in relational settings. Also, several characteristics of DNs are particularly desirable for modeling relational data. First, learning a collection of conditional models offers significant efficiency gains over learning a full joint model—this is generally true, but is even more pertinent to relational settings where the feature space is very large. Second, networks that are easy to interpret and understand aid analysts' assessment of the utility of the relational information. Third, the ability to represent cycles in a network facilitates reasoning with relational autocorrelation, a common characteristic of relational data. Finally, while the need for approximate inference is a disadvantage of DNs for propositional data, due to the complexity of relational model graphs in practice, all PRMs use approximate inference.

RDNs extend DNs for relational data in the same way that RBNs [5] extend Bayesian networks and RMNs [15] extend Markov networks. We describe the general characteristics of PRMs and then discuss the details of RDNs.

3.1. Probabilistic Relational Models

PRMs represent a joint probability distribution over a relational dataset. When modeling attribute-value data with graphical models, there is a single graph G that is associated with the model M. In contrast, there are three graphs associated with models of relational data: the *data graph* G_D, the *model graph* G_M, and the *inference graph* G_I.

First, the relational dataset is represented as a typed, attributed graph $G_D = (V_D, E_D)$. For example, consider the data graph in figure 1a. The nodes V_D represent objects in the data (e.g., authors, papers) and the edges E_D represent relations among the objects (e.g., author-of, cites). We use rectangles to represent objects, circles to represent random variables, dashed lines to represent relations, and solid lines to represent probabilistic dependencies. Each node $v_i \in V_D$ is associated with a type $T(v_i) = t_{v_i}$ (e.g., *paper*). Each object type $t \in T$ has a number of associated attributes $\mathbf{X^t} = (X_1^t, ..., X_n^t)$ (e.g., topic, year). Consequently, each object v_i is associated with a set of attribute values determined by its type $\mathbf{X_{v_i}^{t_{v_i}}} = (X_{v_i1}^{t_{v_i}}, ..., X_{v_in}^{t_{v_i}})$. A PRM model represents a joint distribution over the values of the attributes throughout the data graph, $\mathbf{x} = \{\mathbf{x_{v_i}^{t_{v_i}}} : v_i \in V, t_{v_i} = T(v_i)\}$.

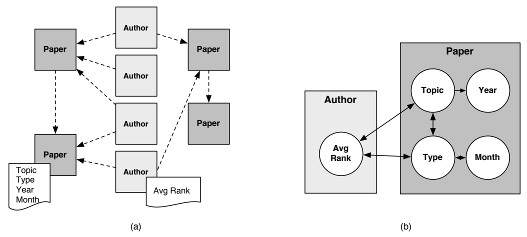


Figure 1. PRM (a) data, and (b) model graph.

The dependencies among attributes are represented in the model graph $G_M = (V_M, E_M)$. Attributes of an object can depend probabilistically on other attributes of the same object, as well as on attributes of other related objects in G_D. For example, the topic of a paper may be influenced by attributes of the authors that wrote the paper. Instead of defining the dependency structure over attributes of specific objects $\mathbf{X_v}$, PRMs define a generic dependency structure at the level of object types. The set of attributes $\mathbf{X_k^t} = (X_{v_ik}^t : v_i \in V,\ T(v_i) = t)$ is tied together and modeled as a single variable. Each node $v_i \in V_M$ corresponds to an $X_k^t, t \in T \wedge X_k^t \in \mathbf{X^t}$. As in conventional graphical models, each node is associated with a probability distribution conditioned on the other variables. Parents of X_k^t are either: (1) other attributes associated with type t_k (e.g., paper *topic* depends on paper *type*), or (2) attributes associated with objects of type t_j where objects t_j are related to objects t_k in G_D (e.g., paper *topic* depends on author *rank*). For the latter type of dependency, if the relation between t_k and t_j is one-to-many, the parent consists of a set of attribute values (e.g., author ranks). In this situation, PRMs use aggregation functions, either to map sets of values into single values, or to combine a set of probability distributions into a single distribution.

For example, consider the model graph in figure 1b. It models the data in figure 1a, which has two object types: paper and author. In G_M, each object type is represented by a plate, and each attribute of each object type is represented as a node. The edges of G_M characterize the dependencies among the attributes at the type level.

During inference, a PRM uses the G_M and G_D to instantiate an inference graph $G_I = (V_I, V_E)$ in a process sometimes called "rollout". The rollout procedure used by PRMs to produce the G_I is nearly identical to the process used to instantiate models such as hidden Markov models (HMMs), and conditional random fields (CRFs) [9]. G_I represents the probabilistic dependencies among all the object variables in a single test set (here G_D is different from the G'_D used for training). The structure of G_I is determined by both G_D and G_M—each object-attribute pair in G_D gets a separate, local copy of the appropriate CPD from G_M. The relations in G_D constrain the way that G_M is rolled out to form G_I. PRMs can produce inference graphs with wide variation in overall and local structure, because the structure of G_I is determined by the specific data graph, which typically has non-uniform structure. For example, figure 2 shows the PRM from figure 1b rolled out over a data set of three authors and three papers, where P_1 is authored by A_1 and A_2, P_2 is authored by A_2 and A_3, and P_3 is authored by A_3. Notice that there is a variable number of authors per paper. This illustrates why PRM CPDs must aggregate—for example, the CPD for paper-type must be able to deal with a variable number of author ranks.

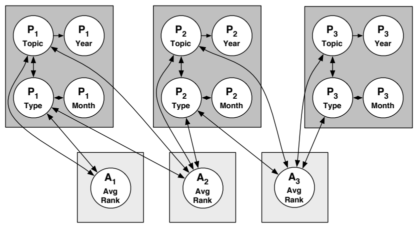


Figure 2. Example PRM inference graph.

3.2. RDN Models

An RDN model encodes probabilistic relationships in a similar manner to DN models, extending the representation to a relational setting. RDNs use a bidirected model graph G_M with a set of conditional probability distributions P. Each node $v_i \in V_M$ corresponds to an $X_k^t \in \mathbf{X^t}$, $t \in T$ and is associated with a conditional distribution $p(x_k^t \mid pa_{x_k^t})$. Figure 1b illustrates an example RDN model graph for the data graph in figure 1a. The graphical representation illustrates the qualitative component (G_D) of the RDN—it does not depict the quantitative component (P) of the model, which includes aggregation functions. The representation uses a modified plate notation; dependencies among attributes of the same object are contained inside the rectangle and arcs that cross the boundary of the rectangle represent dependencies among attributes of related objects. For example, *month*$_i$ depends on *type*$_i$, while *avgrank*$_j$ depends on the *type*$_k$ and *topic*$_k$ for all papers k related to author j in G_D. Although conditional independence is infered using an undirected view of the graph, bidirected edges are useful for representing the set of variables in each CPD. For example, in figure 1b, the CPD for *year* contains *topic* but the CPD for *topic* does not contain *type*. This shows inconsistencies that may result from the RDN learning technique.

Learning. The RDN learning algorithm is much like the DN learning algorithm, except we use a selective relational classification algorithm to learn a set of conditional models. The algorithm input consists of: (1) a data graph G_D, with a set of types T and attributes X, (2) a conditional relational learner R, and (3) a search limit c, which limits the length of paths in G_D that are considered in R. For each $t \in T$, and each $X_k^t \in \mathbf{X^t}$, the algorithm uses R to learn a CPD for X_k^t given the set of attributes $\{X_{k' \neq k}^t\} \cup \mathbf{X^{t' \neq t}}$, where t' is up to c links away from t in G_D. The resulting CPDs are included in P and are used to form G_M.

We use relational probability trees (RPTs) [13] for the CPD components of the RDN. The RPT learning algorithm adjusts for biases towards particular features due to degree disparity and autocorrelation in relational data [7, 8]. We have shown that RPTs build significantly smaller trees than other conditional models and achieve equivalent, or better, performance. This characteristic of the RPTs is crucial for learning understandable RDN models. The collection of RPTs will be used during inference so the size of the models also has a direct impact on efficiency. We expect that the general properties of RDNs would be retained if other approaches to learning conditional probability distributions were used instead, given that those approaches are both selective and accurate.

RPTs extend probability estimation trees to a relational setting. RPT models estimate probability distributions over class label values in the same manner as conventional classification trees, but the algorithm looks beyond the attributes of the item for which the class label is defined and considers the effects of attributes in the local relational neighborhood ($\leq c$ links away) on the probability distribution. The RPT algorithm automatically constructs and searches over aggregated relational features to model the distribution of the target variable—for example, to predict the value of an attribute (e.g., paper topic) based on the attributes of related objects (e.g., characteristics of the paper's references), a relational feature may ask whether the oldest reference was written before 1980.

Inference. The RDN inference graph G_I is potentially much larger than the original data graph. To model the full joint distribution there must be a separate node (and CPD) for each attribute value in G_D. To construct G_I, the set of template CPDs in P is rolled-out over the data graph. Each object-attribute pair gets a separate, local copy of the appropriate CPD. Consequently, the total number of nodes in the inference graph will be $\sum_{v \in V_D} |\mathbf{X^{T(v)}}|$. Rollout facilitates generalization across data graphs of varying size—we can learn the CPD templates from one data graph and apply the model to a second data graph with a different number of objects by rolling out more CPD copies.

We use Gibbs sampling (e.g. [11]) for inference in RDN models. To estimate a joint distribution, the inference graph consists of a rolled-out network with unobserved variables. The values of all unobserved variables are initialized to values drawn from their prior distributions. Gibbs sampling then iteratively relabels each unobserved variable by drawing from its local conditional distribution, given the current state of the rest of the graph. After a sufficient number of iterations, the values will be drawn from a stationary distribution and we can use the samples to estimate probabilities of interest. For the experiments reported in this paper we use a fixed-length chain of 2000 samples (each iteration relabels every value sequentially), with a burn-in of 200.

4. Experiments

The experiments in this section are intended to demonstrate the utility of RDNs as a joint model of relational data. We learn RDN models of four real world datasets to illustrate the types of domain knowledge that can be garnered. In addition, we evaluate the models in a classification context, where only a single attribute is unobserved in the test set, and report significant performance gains compared to a conditional model. Finally, we use synthetic data to assess the impact of training set size and autocorrelation on RDN learning and inference, showing that accurate models can be learned at small data set sizes and that the model is robust to all but extreme levels of autocorrelation. For these experiments, we used the parameters $R = RPT$ and $c = 2$. The RPT algorithm used *MODE*, *COUNT* and *PROPORTION* features with 10 thresholds per attribute.

The RDN models in figures 3-5 continue with the RDN representation introduced in figure 1. Each object type is represented in a separate plate, arcs inside a plate indicate dependencies among the attributes of a single object and arcs crossing the boundaries of plates indicate dependencies among attributes of related objects. An arc from x to y indicates the presence of one or more features of x in the RPT learned for y.

When the dependency is on attributes of objects more than a single link away, the arc is labeled with small rectangle to indicate the intervening object type. For example, movie genre is influenced by the genres of other movies made by the movie's director, so the arc would be labeled with a small *D* rectangle.

In addition to dependencies among attribute values, RPTs also learn dependencies between the structure of the relations (edges in G_D) and the attribute values. This *degree* relationship is represented by a small black circle in the corner of each plate, arcs from this circle indicate a dependency between the number of related objects and an attribute value of a related object. For example, movie receipts is influenced by the number of actors in the movie.

4.1. RDN Models

The first data set is drawn from the Internet Movie Database (www.imdb.com). We collected a sample of 1,382 movies released in the United States between 1996 and 2001. In addition to movies, the data set contains objects representing actors, directors, and studios. In total, this sample contains approximately 42,000 objects and 61,000 links. We learned a RDN model for ten discrete attributes including actor gender and movie opening weekend receipts ($>$*\$2million*). Figure 3 shows the resulting RDN model. Four of the attributes, movie receipts, movie genre, actor birth year, and director 1^{st} movie year, exhibit autocorrelation dependencies. Exploiting this type of dependency has been shown to significantly improve classification accuracy of RMNs compared to RBNs which cannot model cyclic dependencies [15]. However, to exploit autocorrelation the RMN must be instantiated with a corresponding clique template—the dependency must be pre-specified by the user. To our knowledge, RDNs are the first PRM capable of *learning* this type of autocorrelation dependency.

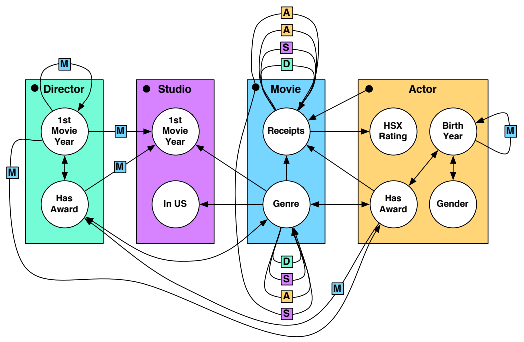


Figure 3. Internet Movie database RDN.

The second data set is drawn from Cora, a database of computer science research papers extracted automatically from the web using machine learning techniques [10]. We selected the set of 4,330 machine-learning papers along with associated authors, cited papers, and journals. The resulting collection contains approximately 13,000 objects and 26,000 links. We learned an RDN model for seven attributes including paper topic (e.g., neural networks) and journal name prefix (e.g., IEEE). Figure 4 shows the resulting RDN model. Again we see that four of the attributes exhibit autocorrelation. In particular, notice that the topic of a paper depends not only on the topics of other papers that it cites, but also on the topics of other papers written by the authors. This model is a good reflection of our domain knowledge about machine learning papers.

The third data set was collected by the WebKB Project [4]. The data consist of a set of 3,877 web pages from four computer science departments. The web pages have been manually labeled with the categories: course, faculty, staff, student, research project, or other. The collection contains approximately 4,000 web pages and 8,000 hyperlinks among those pages. We learned an RDN model for four attributes of the web pages including school and page label. Figure 5a shows the resulting RDN model.

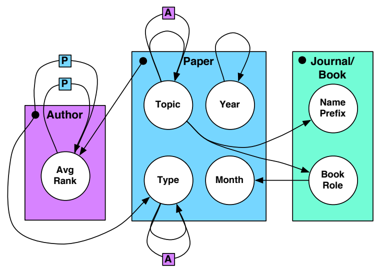


Figure 4. Cora machine-learning papers RDN.

The fourth data set is a relational data set containing information about the yeast genome at the gene and the protein level (www.cs.wisc.edu/∼dpage/kddcup2001/). The data set contains information about 1,243 genes and 1,734 interactions among their associated proteins. We learned an RDN model for seven attributes. The attributes of the genes included protein localization and function, and the attributes on the interactions included type and level of expression. Figure 5b shows the resulting RDN model.

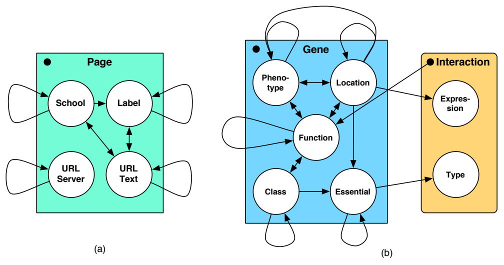


Figure 5. (a) WebKB, and (b) gene data RDNs.

4.2. Classification Experiments

We evaluate the learned models on classification tasks in order to assess (1) whether autocorrelation dependencies among instances can be used to improve model accuracy, and (2) whether the RDN models, using Gibbs sampling, can effectively infer labels for a network of instances. To do this, we compare three models. The first model is a conventional RPT model—an individual classification model that reasons about each instance independently from other instances and thus does not use the class labels of related instances. The second model is a RDN model that exploits additional information available in labels of related instances and reasons about networks of instances collectively. The third model is a probabilistic ceiling for the RDN model. We use the RDN model but allow the true labels of related instances to be used during inference. This model shows the level of performance possible if the RDN model could infer the true labels of related instances with perfect accuracy.

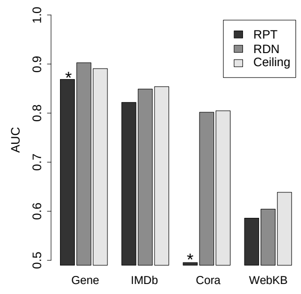


Figure 6. AUC results for classification tasks.

Figure 6 shows area under the ROC curve (AUC) results for each of the three models on four classification tasks. We used the following prediction tasks: IMDb: movie receipts, Cora: paper topic, WebKB: page label, Gene: gene location. The graph shows AUC for the most prevalent class, averaged over a number of training/test splits. For IMDb and Cora, we used 4-5 temporal samples where we learned models on one year of data and applied the model to the subsequent year. For WebKB, we used cross-validation by department, learning on three departments and testing on pages from the fourth held out department. For Gene there was no clear sampling choice, so we used ten-fold cross validation on random samples of the genes. We used two-tailed, paired t-tests to assess the significance of the AUC results obtained from the trials. The t-tests compare the RDN results to each of the other two models. The null hypothesis is that there is no difference in the AUC results of the two models; the alternative is that there is a difference. The differences in performance that are significant at a $p < 0.05$ level are reported in the graph with asterisks.

On three of the tasks, the RDNs models achieve comparable performance to the ceiling models and on the fourth (WebKB) the difference is not statistically significant. This indicates that the RDN model realized its full potential, reaching the same level of performance as if it had access to the true labels of related movies. On the Gene data, the RDN surpasses the performance of the ceiling model, but is only a probabilistic ceiling—the RDN may perform better if an incorrect prediction for one object improves the classification of related objects. Also, the performance of the RDN models is superior to RPT models on all four tasks. This indicates that autocorrelation is both present in the data and identified by the RDN models. The performance improvement over RPTs is due to successful exploitation of this autocorrelation. On the Cora data, the RPT model performance is no better than random because autocorrelation is the only predictor of paper topic (see figure 4).

4.3. Synthetic Data Experiments

To explore the effects of training set size and autocorrelation on RDN learning and inference, we generated homogeneous data graphs with a regular square lattice structure. With the exception of objects along the outer boundary, each object in the lattice links to four immediate neighbors positioned above, below, left, and right. The first and last row and column make up the *frame* of the lattice. In order to control for effects of varying structure, objects in the frame are not used during learning and inference, although their attribute values are available to objects in the core for learning and inference. Thus, training or test sets of size N correspond to a lattice of $(\sqrt{N}+2)^2$ objects, and models are trained or evaluated on the N objects in the core of the lattice. Each object has four boolean attributes X_1, X_2, X_3 and X_4. We use a simple RDN where X_1 is autocorrelated (through objects one link away), X_2 depends on X_1, and the other two attribute have no dependencies.

We generated the values of attributes using the RDN in the following way. We begin by assigning each object in the lattice an initial value for X_1 with $P(x_1 = 1) = 0.5$. We then perform Gibbs sampling over the entire lattice to estimate the values of X_1 conditioned on neighboring values. The values assigned to each object after 200 iterations are used as the final labels. We use a manually specified RPT that assigns X_1 values to each object based on the X_1 values of objects one link away in G_D. The parameters of this model are varied to produce different levels of autocorrelation in X_1. Once X_1 values are assigned, values for X_2 are randomly drawn from a distribution conditioned on objects' X_1 values. We used the parameters $p(x_2 = 1) = 0.3$ and $p(x_1 = 1|x_2 = 1) = 1 - p(x_1 = 0|x_2 = 1) = 0.9$. Finally, random values are assigned to the two other attributes with $p(x_3 = 1) = p(x_4 = 1) = 0.5$. Once a dataset is generated, we measure the proportion of objects with $X_1 = 1$, and any dataset with a value outside the range $[0.4, 0.6]$ is discarded and replaced by a new dataset. This ensures consistency in the distribution of X_1 across datasets and reduces variance in estimated model performance.

The first set of synthetic experiments examines the effectiveness of the RDN learning algorithm. Figure 7a graphs the log-likelihood of learned models as a function of training set size. Training set size was varied at the following levels $\{25, 49, 100, 225, 484, 1024, 5041\}$. Figure 7b graphs log-likelihood as a function of autocorrelation. Autocorrelation was varied to approximate the following levels $\{0.0, 0.25, 0.50, 0.75, 1.0\}$. (We graph the average autocorrelation for each set of trials, which is within 0.02 of these numbers.) At each data set size (autocorrelation level), we generated 25 training sets and learned RDNs. Using each learned model, we measured the average log-likelihood of another 25 test sets (size 225). Figure 7 plots these measurements as well as the log-likelihood of the test data from the RDN used for data generation. These experiments show that the learned models are a good approximation to the true model by training set size 1000, and that RDN learning is robust with respect to varying levels of autocorrelation.

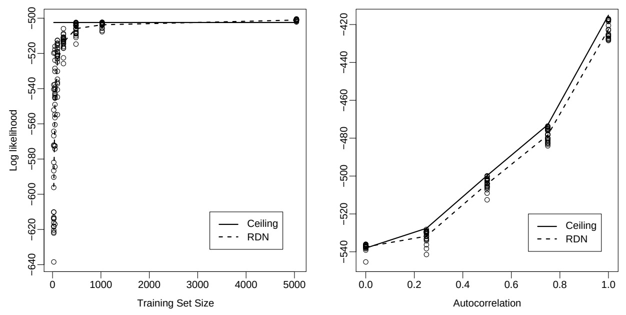


Figure 7. Evaluation of RDN learning.

The second set of synthetic experiments evaluates the RDN inference procedure in a classification context, where only a single attribute is unobserved in the test set. We generated data in the manner described above, learned an RDN for X_1, used the learned models to infer the class labels of unseen test sets, and measured AUC to evaluate the predictions. These experiments compared the same three models as section 4.2, and used the same training set sizes and autocorrelation levels outlined above. At each data set size (autocorrelation level), we generated 25 training and test set pairs, learned the model on the training set, and inferred labels for the test set.

Figure 8a graphs AUC as a function of training set size for RDNs compared to RPTs and the ceiling, plotting the average AUC for each model type. Even at small data set sizes the RDN performance is close to optimal and significantly higher than the performance of the RPTs. Surprisingly, the RPTs are able to achieve moderately good results even without the class labels of related instances. This is because the RPTs are able to use the attribute values of related instances as a surrogate for autocorrelation.

Figure 8b plots average AUC as a function of autocorrelation for RDNs compared to RPTs and the ceiling. When there is no autocorrelation, the RPT models perform optimally. In this case, the RDNs are slightly biased due to excess structure. However, as soon as there is minimal autocorrelation, the RDN models start to outperform the RPTs. At the other extreme, when autocorrelation is almost perfect the RDNs experience a large drop in performance. At this level of autocorrelation, the Gibbs sampling procedure can easily converge to a labeling that is "correctly" autocorrelated but with opposite labels. Although all 25 trials appeared to converge, half performed optimally and the other half performed randomly (AUC ≈ 0.50). Future work will explore ways to offset this drop in performance. However, the utility of RDNs for classification is clear in the range of autocorrelations that have been observed empirically [0.25,0.75].

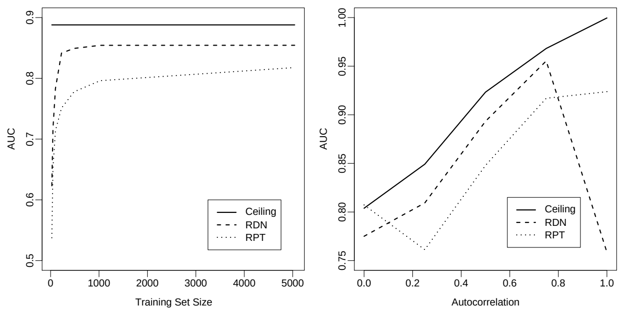


Figure 8. Evaluation of RDN inference.

5. Discussion and Conclusions

In this paper we presented RDNs, a new form of PRM. The primary advantage of RDN models is the ability to learn and reason with relational autocorrelation. We showed the RDN learning algorithm to be a relatively simple method for learning the structure and parameters of a probabilistic graphical model. In addition, RDNs allow us to exploit existing techniques for learning conditional probability distributions. Here we have chosen to exploit our prior work on RPTs, which constructs parsimonious models of relational data, but we expect that the general properties of RDNs would be retained if other approaches to learning conditional probability distributions were used, given that those approaches are both selective and accurate.

The results of the real and synthetic data experiments indicate that collective classification with RDNs can offer significant improvement over conditional approaches to classification when autocorrelation is present in the data—a nearly ubiquitous characteristic of relational datasets. The performance of RDNs also approaches the performance that would be possible if all the class labels of related instances were known. Future work will compare RDN models to RMN models in order to better assess the quality of the pseudolikelihood approximation of the joint distribution. In addition, we are exploring improved inference procedures that consider the autocorrelation dependencies in the data in order to improve inference accuracy and efficiency.

Acknowledgments

The authors acknowledge the invaluable assistance of M. Cornell, A. Shapira, M. Rattigan, M. Hay, B. Gallagher, and helpful comments from A. McGovern, A. Fast, C. Loiselle, and S. Macskassy. This research is supported under a AT&T Labs Graduate Research Fellowship and by NSF and AFRL under contract numbers F30602-00-2-0597, EIA9983215 and F30602-01-2-0566.

References

[1] J. Besag. Statistical analysis of non-lattice data. *The Statistician*, 24:3:179–195, 1975.

[2] W. Buntine. Graphical models for discovering knowledge. In *Advances In Knowledge Discovery And Data Mining*. AAAI Press/The MIT Press, 1996.

[3] S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 307–318, 1998.

[4] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to extract symbolic knowledge from the world wide web. In *Proceedings of the 15th National Conference on Artificial Intelligence*, pages 509–516, 1998.

[5] L. Getoor, N. Friedman, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *Relational Data Mining*. Springer-Verlag, 2001.

[6] D. Heckerman, D. Chickering, C. Meek, R. Rounthwaite, and C. Kadie. Dependency networks for inference, collaborative filtering and data visualization. *Journal of Machine Learning Research*, 1:49–75, 2000.

[7] D. Jensen and J. Neville. Linkage and autocorrelation cause feature selection bias in relational learning. In *Proceedings of the 19th International Conference on Machine Learning*, pages 259–266, 2002.

[8] D. Jensen and J. Neville. Avoiding bias when aggregating relational data with degree disparity. In *Proceedings of the 20th International Conference on Machine Learning*, pages 274–281, 2003.

[9] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 282–289, 2001.

[10] A. McCallum, K. Nigam, J. Rennie, and K. Seymore. A machine learning approach to building domain-specific search engines. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 662–667, 1999.

[11] R. Neal. Probabilistic inference using markov chain monte carlo methods. Technical Report CRG-TR-93-1, Dept of Computer Science, University of Toronto, 1993.

[12] J. Neville and D. Jensen. Collective classification with relational dependency networks. In *Proceedings of the 2nd Multi-Relational Data Mining Workshop, KDD2003*, pages 77–91, 2003.

[13] J. Neville, D. Jensen, L. Friedland, and M. Hay. Learning relational probability trees. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 625–630, 2003.

[14] S. Sanghai, P. Domingos, and D. Weld. Dynamic probabilistic relational models. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pages 992–1002, 2003.

[15] B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*, pages 485–492, 2002.

Autocorrelation and Relational Learning: Challenges and Opportunities

Jennifer Nevillle JNEVILLE@CS.UMASS.EDU
Özgür Şimşek OZGUR@CS.UMASS.EDU
David Jensen JENSEN@CS.UMASS.EDU
University of Massachusetts, Amherst MA 01003-9264 USA

Abstract

Autocorrelation, a common characteristic of many datasets, refers to correlation between values of the same variable on related objects. It violates the critical assumption of instance independence that underlies most conventional models. In this paper, we provide an overview of research on autocorrelation in a number of fields with an emphasis on implications for relational learning, and outline a number of challenges and opportunities for model learning and inference.

1. Introduction

Autocorrelation refers to correlation between values of the same variable on related objects. More formally, it is defined with respect to a set of related instance pairs $(z_i, z_j) \in Z$ and a variable X defined on these instances, and is the correlation between the values of X on these instance pairs. Autocorrelation is a common characteristic of many datasets. For example, hyperlinked web pages are more likely to share the same topic than randomly selected pages (Taskar et al., 2002), and proteins located in the same place in a cell (e.g., mitochondria or cell wall) are more likely to share the same function (e.g., transcription or cell growth) than randomly selected proteins (Neville & Jensen, 2002).

The prevalence of autocorrelation is not unexpected—a number of widely occurring phenomena give rise to such dependencies. Temporal and spatial locality very often result in autocorrelated observations, due to temporal or spatial dependence of measurement errors, or the existence of a variable whose influence is correlated among instances that are located closely in time or space (Mirer, 1983; Anselin, 1998). Social phenomena such as social influence (Marsden & Friedkin, 1993), diffusion processes (Doreian, 1990), and the principle of homophily (McPherson et al., 2001) give rise to autocorrelated observations as well, through their influence on social interactions that govern the data generation process.

Presence of autocorrelation is a strong motivation for relational learning and inference. It is well known that in relational domains, joint inference over an entire dataset results in more accurate predictions than conditional inference over each instance independently (Macskassy & Provost, 2003; Chakrabarti et al., 1998; Taskar et al., 2002; Yang et al., 2002; Neville & Jensen, 2003). Recent work has shown that the improvement over conditional models increases with increased autocorrelation (Jensen et al., 2004)—autocorrelation allows inferences on one object to be useful for inferences on related objects.

The presence of autocorrelation, however, also presents additional challenges for learning. A major difficulty is that the assumption of independent data instances that underlie most conventional models is no longer valid. For instance, in models constructed from temporal and spatial datasets, autocorrelation has long been recognized as a source of increased bias and variance (Anselin, 1998). These problems are only more severe in relational data that do not exhibit the regularities of temporal and spatial datasets. For example, *linkage*—a measure of the number of related instances—can be far greater and can vary dramatically throughout the dataset, and it is known that linkage interacts with autocorrelation to increase variance and such variance can bias feature selection toward features with the least amount of evidence (Jensen & Neville, 2002).

Datasets exhibiting autocorrelation are common in many fields including sociology, economics, geography, and physics (Doreian, 1990). Social network analysis often examines networks of social interactions which exhibit homophily. For example, in elementary school friendship networks, same-gender ties are more likely than different-gender ties (Anderson et al., 1999). Economic analysis often examines datasets with repeated measures of the same variable over time, which typically exhibit temporal autocorrelation (e.g., stock prices). As a consequence, researchers in these fields have investigated the effects of autocorrelation in parameter estimation, hypothesis testing, and structure search.

A common finding in these disparate fields is that departures from independence cannot be ignored—they may cause unduly complex models, and biased, inconsistent, or inefficient estimators. One possible approach is to design new statistical procedures that are robust to autocorrelation. A second one is to model dependencies explicitly.

In this paper, we provide an overview of research on autocorrelation in these fields with an emphasis on implications for machine learning. The remainder of this paper is organized as follows: First, we provide an overview of work in temporal sequence analysis focusing on work in econometrics. This field has a long history of analyzing the effects of autocorrelation. We next discuss research in spatial statistics that extend one-dimensional temporal models to address the needs of higher-dimensional spatial data, and continue with work in social network analysis on general network data. We then briefly outline models in relational learning and discuss the utility and implications of work in related fields for relational learning models.

2. Temporal Sequential Models

Linear regression models are commonly employed in both natural and social sciences to model the dependence of a single response variable Y on a set of predictor variables $\mathbf{X} = \{X^1, \ldots, X^m\}$. The conventional linear regression model is specified as follows:

$$Y_i = \beta \mathbf{X_i} + \epsilon_i \qquad (1)$$

where β is a vector of weights, ϵ is a normally-distributed error term with mean 0, and i is an index over data instances. The weight vector β is usually estimated using Ordinary Least Squares (OLS), which is known to be the Best Linear Unbiased Estimator (BLUE)—the minimum variance estimator for the class of linear unbiased estimators.

One of the implicit assumptions underlying these models is that instances are independent. However, this assumption is violated in many datasets consisting of observations over time. For example, the daily closing price of a stock market index (e.g., S&P500) can be represented as a time series. It is well known that stock prices exhibit autocorrelation over time—the best prediction of tomorrow's stock prices is based on today's prices (Wooldrige, 2003). [1]

If a conventional linear regression model is used to model autocorrelated data, the residuals of the model will be autocorrelated. This violates the modeling assumption of independent and identically distributed errors. For example, if equation 1 is used to regress a number of market indicators $\mathbf{X}$ (e.g. unemployment rate, federal interest rate) on the index price Y, errors will be similar for instances close in time due to the autocorrelation of Y. Serially correlated errors can be detected using a variety of statistics. The most widely-used is the Durbin-Watson statistic, which is a normalized sum of the squared differences of successive terms in a time series (Kennedy, 1998).

When the errors are autocorrelated, OLS estimators are unbiased, but they are no longer BLUE (Wooldrige, 2003). That is, there exist other unbiased linear estimators with lower variance. Not accounting for the autocorrelation structure results in larger sampling errors for the β estimates. Typically, this increased variance will bias hypothesis tests in the direction of increased Type I errors (rejecting the null hypothesis when it is true) and will result in incorrect conclusions of significance. Furthermore, the amount of bias will increase as the level of autocorrelation increases.

Autocorrelated errors typically arise in one of two situations. First, autocorrelated errors may be due to correlated measurement errors. For example, trading patterns can produce serially correlated estimates of stock returns even when there is no serial correlation for returns in general. Returns are measured using the price of the stock on the last trade in a given time period; if the measurement time period is short and the stock is sparsely traded, the estimates of return values will exhibit autocorrelation. Models that represent such autocorrelation dependencies among error terms are known generally as *disturbances models*, but are also referred to as heterogeneity models in spatial analysis, or as serial correlation models in temporal analysis. Second, autocorrelated errors may be due to correlation of the response values. For example, as was mentioned above, the price of an index today may

[1] Unfortunately, this characteristic cannot be used for accurate prediction because the chance of a stock's future price going up is the same as it going down. The overall process is is a random walk.

influence the price tomorrow. This case is typically modeled by including a lagged value of the response variable as a regressor. Models that represent these dependencies are known generally as *effects models*, but are also referred to as autoregressive models or as dependence models in spatial and social network analysis. Below we discuss each of these in turn.

2.1. Disturbances Model

Serial correlation implies that there is systematic dependence among the error terms of individual instances. The most common form is first-order serial correlation, in which the error term in one period includes a proportion of the error term in the previous period. The is commonly referred to as an AR(1) disturbance model:

$$Y_i = \beta \mathbf{X_i} + \mu_i, \text{ where } \mu_i = \rho\mu_{i-1} + \epsilon_i \qquad (2)$$

where ρ is a parameter called the autocorrelation coefficient, whose absolute value is constrained to be less than 1. When $\rho = 0$, this model reduces to the standard linear model of equation 1.

When serial autocorrelation is present, analysts generally abandon OLS in favor of Generalized Least Squares (GLS) estimators that are BLUE. Unfortunately, knowledge of the correlation structure is needed for exact GLS estimates and in general this is not known apriori. Alternative Estimated Generalized Least Squares (EGLS) methods estimate ρ and β iteratively or jointly. EGLS estimators are neither linear nor unbiased but Monte Carlo studies have shown that EGLS is preferable to OLS in many situations (Kennedy, 1998). In particular, for the AR(1) disturbances model, EGLS is equal, or superior, to OLS when $\rho > 0.3$. The most frequently used EGLS methods are Cochrane-Orcutt iterative least squares, Durbin's two-stage method, Hildreth-Lu search procedure, and maximum likelihood. These four methods mainly differ in how they estimate ρ and are asymptotically equivalent if ϵ is distributed normally. Recent studies have shown that Bayesian estimation, which averages over a number of ρ estimates, is far superior to methods that use a single estimate (Kennedy, 1998).

2.2. Effects Model

Effects models take into account dependencies among the response values by including a lagged value of the response variable as one of the regressor variables. When lag equals 1 (e.g. first-order autocorrelation), the underlying model is referred to as an AR(1) effect model:

$$Y_i = \rho Y_{i-1} + \beta \mathbf{X_i} + \epsilon_i \qquad (3)$$

When the underlying process is correctly modeled with equation 3, OLS estimators are biased but consistent as long as the errors are *contemporaneously* uncorrelated. This means that the n^{th} regressor term is not correlated with the n^{th} error term; it may be correlated with other error terms. In this case, analysts consider OLS to be the most appropriate estimator (Kennedy, 1998). In small samples, the OLS estimate for ρ is downward-biased, and the OLS estimate for β is upward-biased. In general however, there are no other estimators with superior small-sample properties so analysts prefer OLS for its asymptotic properties. Research has focused on obtaining unbiased OLS estimates for a range of specific autoregressive models, with recent work proposing a Monte Carlo based approach for models with non-normal error terms, higher-order autocorrelations, and exogenous variables (Tanizaki, 2000).

If, on the other hand, the errors are contemporaneously correlated, OLS estimators are biased and inconsistent. A two-step EGLS (as described in section 2.1) is not feasible in this situation because the residuals are correlated with the exogenous variables. The most common approach to take in this situation is instrumental variable (IV) estimation, which introduces extra *instument* variables to decouple the correlation between the regressors and the error terms to produce consistent estimators.

Autoregressive conditional heteroskedasticity (ARCH) models extend the basic AR models described above to model volatility clustering with non-constant variance that depends on past information (Engle, 1995). If the model does not include lagged-dependent variables, OLS estimators are BLUE, but non-linear maximum likelihood estimators are more efficient. If the model includes lagged-dependent variables then the OLS standard errors will not be consistent. In this case, EGLS estimators are asymptotically efficient and standard errors are asymptotically valid.

3. Spatial Models

Spatial datasets are analyzed in a number of fields including geography, biology, and economics. These datasets are typically represented in discrete or continuous two-dimensional space. For example, a spatial dataset may record soil properties throughout a spatial region. Equation 1 may also be used to model these data, for example to model the effects of soil properties on ground water contamination. In this case each vector index i indicates a point in space. We will focus on (simpler) models for discrete space where the data are represented as a lattice—each point in space corresponds to a node in the graph and is linked to a fixed number of other nodes that are closest with respect to a distance measure.

Tests for the presence of residual spatial autocorrelation are based on either OLS or ML estimates, including tests based on Moran's **I** statistic, and Wald, Likelihood Ratio, and Lagrange Multiplier tests (Anselin, 1998). If spatial data exhibit autocorrelation, the quality of OLS parameter estimates are affected in the same manner as was discussed for temporal data—OLS estimators are unbiased but they are no longer BLUE (Anselin, 1998). Again this results in biased hypothesis tests, with the amount of bias depending on the level of autocorrelation.

Dependencies among instances occur in the same manner as in the temporal model discussed above. Autocorrelated errors may be due to spatially correlated measurement errors. For example, a severe weather event may affect only part of the region, resulting in a cluster of correlated errors. On the other hand, autocorrelated errors may be due to spatial autocorrelation in the response variable itself—contamination levels are likely to correlated with the levels at nearby locations.

3.1. Disturbances Model

When the data exhibit autocorrelated disturbances, the error term of one instance influences the error terms of neighboring instances. A spatial disturbances model subsumes the first-order serial correlation model (equation 2) by allowing more general dependencies among the error terms:

$$Y_i = \beta \mathbf{X_i} + \mu_i, \text{ where } \mu_i = \rho \mathbf{W} \mu + \epsilon_i \tag{4}$$

Here $\mathbf{W}$ is an $n \times n$ weight matrix specifying the nature of dependencies among the disturbances, and ρ is the autocorrelation parameter. When $\rho = 0$ or W is uniformly 0, this model reduces to the standard linear model of equation 1. The matrix $\mathbf{W}$ is designed to represent the influence processes present in the network. Each entry w_{ij} denotes the influence node j has on node i. For example, in a first-order spatial disturbances model, row i has a value of 1 for each neighbor j of node i and all other entries are 0.

Spatial autocorrelation among error terms has been shown to affect the quality of OLS parameter estimates (Anselin, 1998). The effects are similar to those reported for temporal models—OLS estimators will be unbiased but inefficient and GLS estimators are BLUE but are of academic interest only because the correlation structure is generally unknown. Futhermore, the multidirectional nature of spatial dependencies limits the types of EGLS methods that will produce consistent estimates. Approaches based on ML or IV result in consistent estimates of ρ and therefore retain the asymptotic properties of consistency and efficiency. However, in small samples, OLS may sometimes perform equivalently, or better than EGLS, in terms of bias and mean squared error—though finite sample analysis is limited (Anselin, 1998).

3.2. Effects Model

The second type of dependency is again due to autocorrelation of the regressor values. The spatial effects model represents these dependencies with the following:

$$Y_i = \rho \mathbf{W} \mathbf{Y} + \beta \mathbf{X_i} + \epsilon_i \tag{5}$$

Again, when $\rho = 0$ or W is uniformly 0, this model reduces to the standard regression model (equation 1).

If the response variable is autocorrelated, OLS estimators will be biased, inconsistent, and inefficient regardless of the properties of the error term (Anselin, 1998). In temporal effects models (equation 3), the OLS estimates will be unbiased if the error terms show no serial correlation. The multidirectional nature of spatial dependencies however, introduces added complexity to the OLS estimates so the conditions for consistency are only met when autocorrelation is not present, when $\rho = 0$. This means that no consistent estimates can be obtained for OLS procedures, so spatial analogues of EGLS methods are not appropriate.

Maximum likelihood (ML) estimation does not suffer from the same effects that plague OLS estimation so it is the preferred method of estimation among analysts for both the disturbances and the effects model. ML estimators have attractive asymptotic properties—consistency, efficiency, normality—but are more complex and computationally intensive than OLS. We should also note here that the attractive asymptotic properties of ML estimation do not hold uniformly, but are valid under the following conditions: the existence of the log-likelihood function for the parameters, continuous differentiability of the log-likelihood function, boundedness of partial derivatives, positive definiteness and/or non-singularity of covariance matrices, and the finiteness of quadratic forms (Anselin, 1998). Typically, these conditions are satisfied if the spatial interaction structure ($\rho \mathbf{W}$) is non-explosive (i.e., the correlation between y_i and y_{i+d} goes to zero sufficiently "quickly" as $d \to \infty$, where d is graph distance).

Depending on the model form, ML estimation may involve a normalizing constant that is difficult to compute in closed form. For the models discussed above, this involves computing the log-determinant of an $n \times n$ matrix, which requires $O(n^3)$ operations for dense matrices. Research has focused on techniques to make ML estimation more tractable, including pseudolikelihood estimation (Besag, 1975), approximate ML estimation with Markov Chain Monte Carlo (MCMC) methods (Geyer & Thompson, 1992), and closed-form ML methods that avoid direct computation of the determinant (LeSage & Pace, 2001).

Hypothesis tests for ML estimates include the Wald test, the Likelihood Ratio test, and the Lagrange Multiplier test, all of which are based on the optimal asymptotic properties of the ML estimator. The tests are asymptotically equivalent but care must be taken when interpreting the tests on finite samples because some have higher Type I errors and others have higher Type II errors. The relative power of the tests for spatial data is yet to be investigated (Anselin, 1998).

4. Network Models

Spatial models have been applied extensively in the field of social network analysis where data consist of a network of interactions among entities (e.g., people, institutions). Social network datasets are represented as general graphs and differ from temporal and spatial data representations in that they are not restricted to a uniform structure. For example, to model the effects of socio-economic status on voting behavior in a community, income and status would be measured along with friendship ties to other members in the community. A set of nodes representing people and a set of edges representing their friendships forms the network graph. The graph structure varies as each person has a different number of friends. Again equation 1 may be used to model network data. In this case each vector index i indicates a node in the graph.

Spatial autocorrelation models are expressive enough to use as network autocorrelation models. Equation 4 represents a network disturbances model and equation 5 represents a network effects model (Marsden & Friedkin, 1993). In social network models, the weight matrix $\mathbf{W}$ specifies the social influence patterns present in the network and it can affect virtually all of the conclusions drawn from autocorrelation models (Leenders, 2002). Therefore, correct specification of $\mathbf{W}$ is crucial to the utility of the models. In practice, social network analysts do not estimate $\mathbf{W}$. Instead, they specify a $\mathbf{W}$ manually to model specific theories of social influence such as communication and comparison.

Social network models share the same challenges as spatial models—OLS parameter estimates of autocorrelated data will be inefficient and/or biased and inconsistent, and although ML estimates are more robust, they are computationally intensive (Doreian & K. Teuter, 1984). Simulation studies have demonstrated the superiority of ML estimates over a wide range of conditions (Doreian & K. Teuter, 1984). Although social network datasets are not restricted to a uniform structure, unfortunately there appears to be little work in social networks that examines the impact of varying graph structure on parameter estimation and hypothesis tests.

5. Models in Relational Learning

Datasets with more general dependencies than are seen in temporal, spatial, and social network data are commonplace in relational learning. For example, relational data for citation analysis can be represented as a typed, attributed graph, with nodes representing authors, papers and journals, and edges representing citation and published-in relationships. A model of paper *topic* may include attributes of related authors (e.g., speciality) and journals (e.g., prestige). However, an important characteristic of these data is that topic is autocorrelated—the topic of a paper is not independent of the topics of papers that it cites.

Relational data pose a number of additional challenges for model learning and inference. First, relational data often consider more than one type of entity in the same dataset (e.g., papers, authors and references). Second, relational data have complex dependencies, both as a result of direct relations (e.g., research paper references) and through chaining multiple relations together (e.g., papers published in the same journal). Third, relational data have varying structure (e.g., papers have different numbers of authors, references and citations).

Recent research in relational learning has produced several novel types of models to address these issues, including relational Markov network (RMNs) (Taskar et al., 2002), relational Bayesian networks (RBNs) (Friedman et al., 1999), and relational dependency networks (RDNs) (Neville & Jensen, 2004). These three models have the ability to represent and reason with autocorrelation; however, only RMNs and RDNs can reason with arbitrary forms of autocorrelation—RBNs can only reason with acyclic forms of autocorrelation, such as relationships that are structured by temporal constraints (Friedman et al., 1999).

There are two major findings that relate autocorrelation to learning and inference in relational models: that autocorrelation improves joint inference, and that autocorrelation may bias feature selection. We discuss each of these below.

First, several studies have shown that *joint inference* can significantly reduce classification error (Macskassy & Provost, 2003; Chakrabarti et al., 1998; Taskar et al., 2002; Yang et al., 2002; Neville & Jensen, 2003). Joint inference refers to procedures that make simultaneous statistical judgments about the same variables for a set of related data instances. By making inferences about multiple data instances simultaneously, joint inference can exploit autocorrelation in the data—judgments about one instance can be used to improve inferences about related instances. Recent work has shown that the improvement over conditional models, which make inferences in isolation, increases with increased autocorrelation, and in general, a joint inference procedure performs better when higher-order autocorrelation is present or when few labels are known with certainty (Jensen et al., 2004). In conditional models, the utility of modeling autocorrelation depends on whether the values of the autocorrelated attributes are known. Partially labeled datasets are common, but if the known labels do not exhibit autocorrelation, they cannot be used to seed the inferences. Related work shows that the relative advantage of a joint inference procedure over a conditional procedure reduces as the percentage of labeled data increases (Macskassy & Provost, 2003).

Second, recent research has shown that autocorrelation may bias feature selection (Jensen & Neville, 2002). Concentrated linkage and autocorrelation reduce the effective sample size of a data set, thus increasing the variance of parameter estimates (e.g., feature scores) estimated using that set. This reduction in effective sample size parallels the inefficiencies in temporal and spatial estimators. As a consequence, the probability of Type I errors is increased—features formed from objects with high linkage and autocorrelation may be selected as the best feature, even when the features are random. To our knowledge, few current relational learning algorithms adjust for the increased variance in estimation. Specifically, the current instantiation of RDNs use an underlying conditional model which adjusts for this bias, but the current instantiations of RBNs and RMNs do not. Inefficient parameter estimates will impact both selective (e.g., RBN, RDN) and non-selective (e.g., RMN) models. For both types of models, the increased variance may result in overfitting. In addition, the interpretation of feature weights/scores may be more difficult for non-selective models and structure learning may be biased in selective models.

6. Summary and Discussion

Autocorrelation effects have been studied extensively in other fields and it is clear that they cannot be ignored in relational learning. In particular, if the data exhibit autocorrelation, either autocorrelated measurement errors or an autocorrelated response variable, then conventional parameter estimates will be unbiased but will have increased variance. This has implications for (1) model performance, (2) feature rankings, and (3) feature selection. When the model is learned from “small” samples, the increased variance may lead to overfitting and result in lower performance. Although, we typically have “large” datasets in relational learning, as the level of autocorrelation increases so does the variance—the amount of data needed to offset the increased variance may be be larger than we expect. Increased variance will also impact feature rankings (by feature weights/scores), and consequently feature selection. Non-selective models often use feature weights for interpretation (e.g., to identify the most important features), and selective models use feature weights to learn the structure of the model. Both these endeavors will be adversely affected by the increased variance due to autocorrelation.

How can we adjust for autocorrelation in relational models? Below, we summarize past research and discuss options for model representation, learning and inference.

6.1. Representation

The first decision is how to include autocorrelation in the model representation—whether to model autocorrelation directly through variables or indirectly in the error term. This choice corresponds to selection of the effects model, the disturbances model, or some combination of the two, and may be based on the researcher’s hypothesis about the dependencies present in the data. Explicit representation may result in more interpretable models, since the influence of an autocorrelated response variable is clear. However, implicit representation in the error term may be more broadly applicable. This approach could allow the use of existing models without a change of representation, but with only an adjustment for the effects of autocorrelation.

The second decision is how to encode the autocorrelation dependencies. This decision corresponds to the functional form of autocorrelation (e.g., first-order). For the spatial and network models discussed above, this refers to the specification of the weight matrix. For relational models, this usually refers to specification of autocorrelation features. For example, to predict the topic of a web page, we may include a feature that encodes the topics of other hyperlinked pages. While considerable attention has been paid to accurate parameter estimation in temporal and spatial autocorrelation models, it appears that researchers are less concerned with model/feature selection. However, one could imagine searching over a space of autocorrelation specifications to learn the correct structure.

6.2. Learning

The effect of autocorrelation on parameter estimation has been studied extensively. Below is a summary of the findings in temporal, spatial and network analysis:

1. If autocorrelation is ignored:
    - Parameter estimates are computationally efficient.
    - Parameter estimates are unbiased but have increased variance.
    - Hypothesis tests and confidence intervals may be biased.
2. If autocorrelation is modeled:
    - Parameter estimates are computationally complex, but more tractable approximate methods exist.
    - Parameter estimates are asymptotically optimal but may be biased in finite samples.
    - Finite sample comparison is limited. More complex estimation techniques may not always be justified.
    - Hypothesis tests are asymptotically unbiased but the relative power of various tests may vary on finite samples.

Some examples of model parameters in relational learning include clique potentials and feature weights. Results for temporal and spatial analysis indicate that there may be a tradeoff between computational efficiency and accurate parameter estimation. Understanding the effect of varying levels of autocorrelation on parameter estimation for finite samples is an important area for research for the relational learning community.

The effects on parameter estimation will also impact structure learning. Structure learning typically involves feature selection, which corresponds to either explicit or implicit hypothesis testing. It has been shown that autocorrelation can lead to increased Type I errors in hypothesis tests, which may lead to a unfair comparison among different features. The impact of these errors has not been fully explored in relational learning. Initial results indicate that they lead to overly complex models with excess structure, and may degrade model performance (Neville et al., 2003).

6.3. Inference

The literature on spatial, temporal, and social network autocorrelation models does not provide much guidance for inference because it focuses on accurate model learning rather than prediction of unobserved variables. There has, however, been preliminary work in relational learning that suggests joint inference can significantly reduce classification error, and that this reduction increases with autocorrelation. Clearly, this is an area with many open questions—e.g., Can autocorrelation be exploited to improve inference efficiency? How does autocorrelation interact with various inference procedures? How does the amount of labeled data interact with the level of autocorrelation in the dataset to determine the improvement in accuracy obtained by joint inference?

7. Conclusions

Autocorrelation is ubiquitous—datasets exhibiting autocorrelation are found in a range of fields including sociology, economics, geography, and physics—and has been studied extensively. In this paper we presented findings from econometrics, spatial statistics, and social network analysis. A common finding is that ignoring autocorrelation may result in unduly complex models, and biased, inconsistent, or inefficient estimators. The effects of autocorrelation are sometimes addressed by modeling the autocorrelation explicitly, and sometimes by using statistical procedures that are robust to these effects.

For reasons we stated earlier, we expect autocorrelation to have greater impact on relational models than on temporal, spatial, and network models. Although the presence of autocorrelation has been widely reported for relational datasets, there has been little focus on the impact of autocorrelation on model learning and inference. The results we discuss here reveal that this is an important area for future research.

Acknowledgments

The authors acknowledge helpful comments provided by the anonymous reviewers. This research is supported under a AT&T Labs Graduate Research Fellowship and by DARPA, NSF and AFRL under contract numbers F30602-00-2-0597, EIA9983215 and F30602-01-2-0566. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements either expressed or implied, of DARPA, NSF, AFRL, or the U.S. Government.

References

Anderson, C., Wasserman, S., & Crouch, B. (1999). A p* primer: Logit models for social networks. *Social Networks*, *21*, 37–66.

Anselin, L. (1998). *Spatial econometrics: Methods and models*. The Netherlands: Kluwer Academic Publisher.

Besag, J. (1975). Statistical analysis of non-lattice data. *The Statistician*, *24:3*, 179–195.

Chakrabarti, S., Dom, B., & Indyk, P. (1998). Enhanced hypertext categorization using hyperlinks. *Proc of ACM SIGMOD98*.

Doreian, P. (1990). Network autocorrelation models: Problems and prospects. In *Spatial statistics: Past, present, and future*, chapter Monograph 12, pp. 369–389. Ann Arbor Institute of Mathematical Geography.

Doreian, P., & K. Teuter, C. W. (1984). Network autocorrelation models: Some monte carlo results. *Sociological Methods and Research*, *13:2*, 155–200.

Engle, R. (Ed.). (1995). *Arch: Selected readings*. Oxford: Oxford University Press.

Friedman, N., Getoor, L., Koller, D., & Pfeffer, A. (1999). Learning probabilistic relational models. *IJCAI* (pp. 1300–1309).

Geyer, C., & Thompson, E. (1992). Constrained monte carlo maximum likelihood for dependent data. *Journal of the Royal Statistical Society, Series B*, *54:3*, 657–699.

Jensen, D., & Neville, J. (2002). Linkage and autocorrelation cause feature selection bias in relational learning. *Proceedings of the 19th International Conference on Machine Learning*.

Jensen, D., Neville, J., & Gallagher, B. (2004). *Why collective inference improves relational classification* (Technical Report 04-27). University of Massachusetts.

Kennedy, P. (1998). *A guide to econometrics*. Cambridge, Massachusetts: The MIT Press.

Leenders, R. (2002). Modeling social influence through network autocorrelation: Constructing the weight matrix. *Social Networks*, *24*, 21–47.

LeSage, J., & Pace, R. (2001). Spatial dependence in data mining. In *Data mining for scientific and engineering applications*. Kluwer Academic Publishers.

Macskassy, S., & Provost, F. (2003). A simple relational classifier. *2nd Workshop on Multi-Relational Data Mining, KDD-2003*.

Marsden, P., & Friedkin, N. (1993). Network studies of social influence. *Sociological Methods and Research*, *22:1*, 127–151.

McPherson, M., Smith-Lovin, L., & Cook, J. (2001). Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, *27*, 415–445.

Mirer, T. (1983). *Economic statistics and econometrics*. New York: Macmillan Publishing Co.

Neville, J., & Jensen, D. (2002). Supporting relational knowledge discovery: Lessons in architecture and algorithm design. *Data Mining Lessons Learned Workshop, 19th International Conference on Machine Learning*.

Neville, J., & Jensen, D. (2003). Collective classification with relational dependency networks. *2nd Workshop on Multi-Relational Data Mining, KDD-2003*.

Neville, J., & Jensen, D. (2004). *Dependency networks for relational data* (Technical Report 04-28). University of Massachusetts.

Neville, J., Jensen, D., Friedland, L., & Hay, M. (2003). Learning relational probability trees. *9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Tanizaki, H. (2000). Bias correction of olse in the regression model with lagged dependent variables. *Computational Statistics and Data Analysis*, *34*, 495–511.

Taskar, B., Abbeel, P., & Koller, D. (2002). Discriminative probabilistic models for relational data. *Proceedings of UAI-2002*.

Wooldrige, J. (2003). *Introductory econometrics: A modern approach*. South-Western College Pub.

Yang, Y., Slattery, S., & Ghani, R. (2002). A study of approaches to hypertext categorization. *Journal of Intelligent Information Systems*.